MASTER IN COMPUTER SCIENCE

Console operations subsystem concepts in large-scale operating systems

Hazard, Pascal

Award date:
1985

Awarding institution:
University of Namur

INSTITUT D'INFORMATIQUE
FNDP NAMUR

CONSOLE OPERATIONS SUBSYSTEM CONCEPTS IN LARGE-SCALE OPERATING SYSTEMS

MEMOIRE PRESENTE PAR
Pascal Hazard

EN VUE DE L'OBTENTION
DU DIPLOME DE
LICENCIE ET MAITRE
EN INFORMATIQUE

ANNEE ACADEMIQUE
1984/85

Acknowledgements

First of all, I would like to thank Mr. Ramaekers for having accepted to conduct this thesis.

I am grateful to the personnal of Munich-Perlach laboratories D ST SP 211 and 212 for the help and friendship marks they give me during my training among them.

Special thanks to Mr. Clemens and Mr. Seiler. Their cooperation allowed me to understand better the domain of this work.

I am also grateful to Mr. Brison and Mr. Demus to have so well organized my training.

I would finally thank all the other people involved with the elaboration of this presentation.

General Introduction

In the first computer generations, the system operating was completed from one physical console. This kind of devices had (and keeps today) an important feature: its COST schematically made of two components:

1. HARDWARE: the hardware console configuration often combines many separated devices; this makes it far more (eigth times) expensive than an end-user terminal. Furthermore, as some hardware oriented functions imposes locality (direct connection to the CPU), consoles cannot be shared between several computer centers and they represent for each of them an HEAVY FIXED COST .

2. PEOPLE: system operating from consoles requires constant care; sometimes even during nights and weekends so that the operating staff must be important (two or three shifts) and as human resources are expensive...

With the time; the system complexity increasing, the number of messages reaching the console became too big to allow one human operator to react to each of them quickly and correctly. A first reaction consisted to multiply the consoles (and the operators) number in order to reduce each of them control area. Unfortunatly, the MULTIPLE CONSOLES CONFIGURATION still strenghtened the economical problem so that, nowadays, the operating remains faced to the two old problems: the heavy console fixed cost and the operating complexity.

To reach the system complexity in the future, OPERATING AUTOMATION seems to be a relatively secure (is complex software more secure than human?) and surely economical way. The automatic operator will become an important auxiliary for the human operator. It will assume all the computerizable clerical tasks and perhaps more intelligent ones (1) but it will let to its human partner

- manual tasks,
- not formally expressed communications handling,
- complex diagnosis...

---

(1) See YES/MVS, a real-time expert system designed to help operator of large-scale MVS computer systems. Developed recently by IBM in the OPS5 language, YES/MVS handles the many messages that occur in scheduling jobs, reallocating system resources and avoiding bottlenecks in MVS machines.

In any case, some human operating functions will stay alive; in some case, their volume will stay important. Therefore, more and more operations have to be completed from commonplace end-user terminals in order to reduce the number of consoles to the minimum bound to hardware locality requirements.

The newly appeared distribution trend puts a new light on the operating economical problem. On one hand, indeed, the data processing of many organizations will be widely distributed among small agencies connected (one to the other or to a central office) by a network. On the other, manufacturers would wish to adapt their existing large-scale operating systems to mini-computers.

The operating delays those trends because agencies, working with mini-computers, cannot assume their own operations. Their employees have no skill in computer field and even if it was, the hardware cost is too big to be supported by each little agengy.

Automatic operators in the agencies supervised from one REMOTE OPERATING CENTER would economically allow the introduction of mini-computers in so-called "unsophisticated circles" (see tertiary industries like banks).

As we can see, the operating is at the center of many debates today. In the future, concepts like CENTRAL OPERATING FROM COMMONPLACE TERMINALS, AUTOMATIC OPERATING or REMOTE OPERATING will probably be current. In Siemens operating system BS2000, a first step in those directions is completed with a special interface ($CONSOLE applicaton) allowing some operating functions to be assumed by authorized tasks (DCAM applications) which may eventually open a dialog with an end-user terminal (see Omnis [OMNI75]).

Unfortunatly, historical features make the system management, in general, and its operating, in particular, conceptually quite hybride and sometimes not secure. Furthermore, the design deduced from those concepts is influenced by their hybridity as well as by years of maitenance so that people wonder now wether it will be able to be the basis for the 2000 years operating(s). Keeping this question in mind, the first part of this work intends to study the actual appliances.

I do not intend, however, to analyse weaknesses of today's equipment for themselves. Rather, I intend to study possible alternative approaches to the past development efforts. The thesis of this presentation is, indeed, that conceptual requirements should serve as directional guideline for operating system design and maintenance. Effectiveness should replace efficiency, even for operating systems, if we want the system to be reliable and secure and to allow non disruptive growth (so important in the operating field). The second section of this work will try to state criticisms about the current console operations subsystem design, to show how they are related to conceptual lacks and, finally, to propose new concepts definitions for console operations subsystem design.

The general plan of this work is then simple:

1. Section 1: A Case Study, Console operations in BS2000
2. Section 2: New Console Operations concepts for large centralized Computer System.

SECTION 1

A CASE STUDY :

CONSOLE OPERATIONS IN BS 2000

Chapter 1: Which Computer System ?

A computer, alone, is not a system. Computer-based information systems run, indeed, using many different resources such as human, organisational and technical ones. There are numbers of alternatives for structuring those resources. They range from a completely centralized to a distributed system with an almost infinite number of combinations in between.

For our purpose, the system is defined as based on a LARGE CENTRALIZED COMPUTER SYSTEM to which remote users gain access via a more or less complex communication network. Our interest relies on this centralized computer system.

People, software (system and application), the data base and hardware are the basic components that accomplish the data processing operations in such systems. Application software and the data base are very user-oriented, we do not consider them. The three other components are discussed below.

1. Hardware

A centralized computer system is a closed world, often reduced to a single room. It is mainly characterized by a large and powerful CENTRAL PROCESSING UNIT with a high capacity central memory (in our case from the 7500 and 7700 series).

MAGNETIC DISKS provide voluminous space for on-line mass storage, sequential files and temporary data.

MAGNETIC TAPES are frequently used for backup or in batch environment.

Faster and faster PRINTERS and specialized facilities such as plotters are also housed at the central site.

CARDS READERS AND PUNCHERS sometimes still sleep in a corner.

The whole configuration is monitored from one or more CONSOLES

Many scattered terminals are connected to this monolith but controlling such a number of devices places a considerable load on the central mainframe. Hence, a special computer, known as a FRONT-END PROCESSOR is used for interfacing the hybrid environment to the CPU.

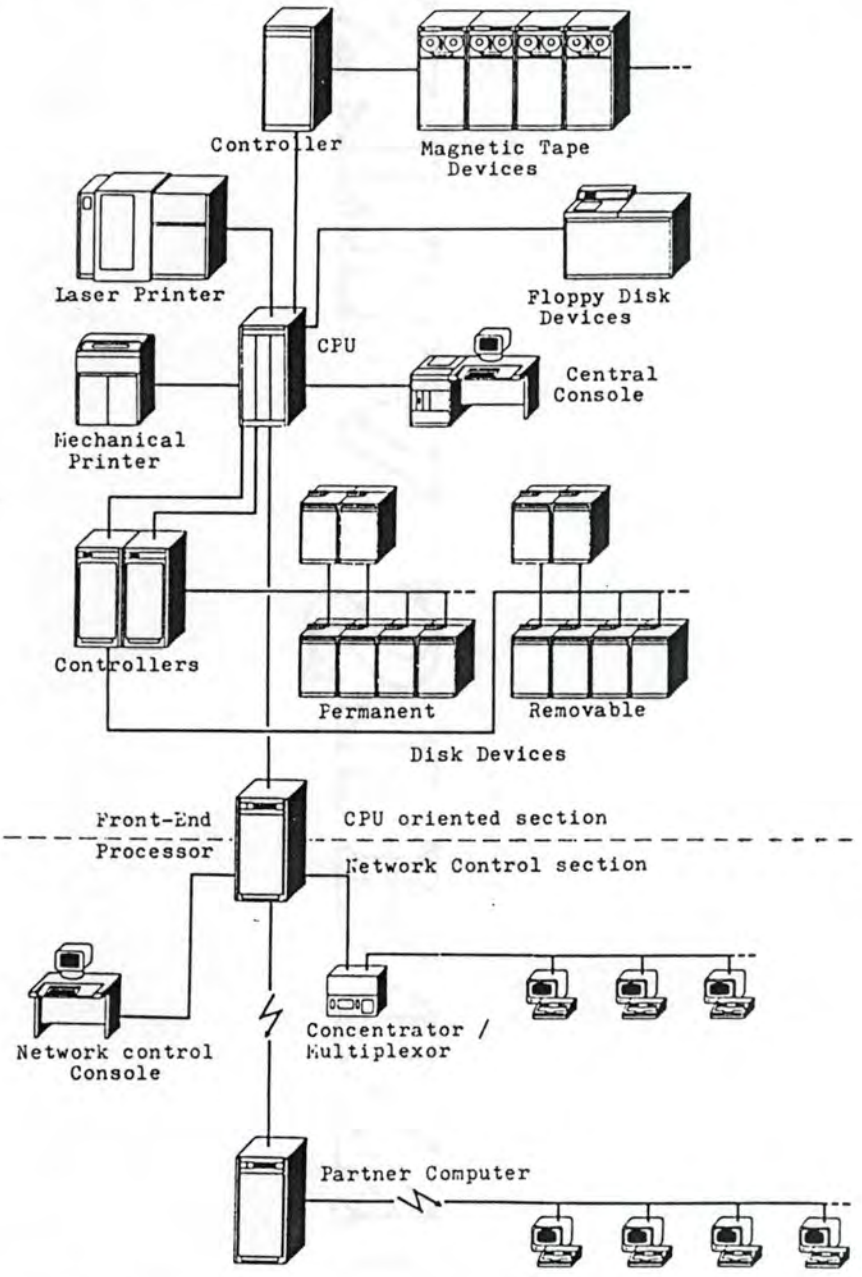
This front-end processor is physically housed in the computer center but it may be visualized logically as consisting of two sections.

One section faces directly the CPU and performs control-type functions similar to other I/O control units. Moreover, being a computer, it can be used to validate data, to collect performance informations or to provide informations for fault diagnostic.

The other section faces the communication network. It is provided

the ability for storing a network control program that allows it to relieve the CPU of the network control task. Indeed, it controls and adapts, via programmed actions,the attached communication channels and terminals or stores data on backing-store for later transmission. (1)

This double nature places the front-end processor sometimes inside, sometimes outside the computer center. We take the logical border between the previously defined sections as the border between the computer center and its environment. Figure 1.1 gives an example of centralized system hardware configuration.



(1) The central mainframe remains the foreman. In particular, it gives the order for the connection or disconnection of the components as well as orders for messages transmission or receipt.

2. The System Software: BS 2000

BS2000 (Betriebsystem 2000) is a large-scale (8MB for version 7.5) operating system derived from the general purpose time-sharing system TSOS developed during the sixties by RCA.

The study of BS2000 by a layman is made difficult by different facts; it is indeed

1. An OLD and COMPLEX system

TSOS was designed before the arising of sofware engineering principles, on a monolithic design. Years or maintainance (the version actually marketted by Siemens is V 7.5) have made BS2000 still more complex. Indeed, consecutive corrections, on one hand; the wish to create more and more sophisticated services, on the other, have led to add new appliances in trying circumstances: on a "wobby" initial design, with respect to strict time imperatives and multiple compatibility constraints (at the user as well as at the hardware level).

2. A MOVING FIELD

New versions succeed quickly one to the other, involving sometimes important design changes. So, by now, versions 8.5 and 9.0 begin their development phase. They will include new concepts like extended architecture, dynamically loaded subsystems, logical consoles....

The following analysis tries to draw up a synthesis about the operating in BS2000 according to informations collected from many sources:

- Lectures at the Schule Fuer Datentechnik (Siemens Munich).
- Development documentation and pieces of code from versions 7.6, 8.0 and 8.5.
- Informal conversations with members of the laboratories involved by operating (D ST SP 211 and 212)

3. People : System Management in BS2000

The managers, well convinced of what people have always repeated to them, -namely that the computer is just a tool- have assigned to the computer department the same finality as to the other organization sectors. Consequently, they apply to it previously defined management principles and many theoretical structures are proposed today for the computer staff.

In practice, however, the computer center is felt as stanger to the rest of the organization. It is led by its own rules applied by a cohort of specialists under the manufacturer's control. Its management stucture is mainly depending on the interface provided by the manufacturer and its associated philosophy (or its lack). therefore, one may speak about system management associated to an operating system.

In figure 1.2, an organization chart illustrates the management philosophy associated with BS2000. The categories shown are administration, operating, data preparation and hardware maintenance.

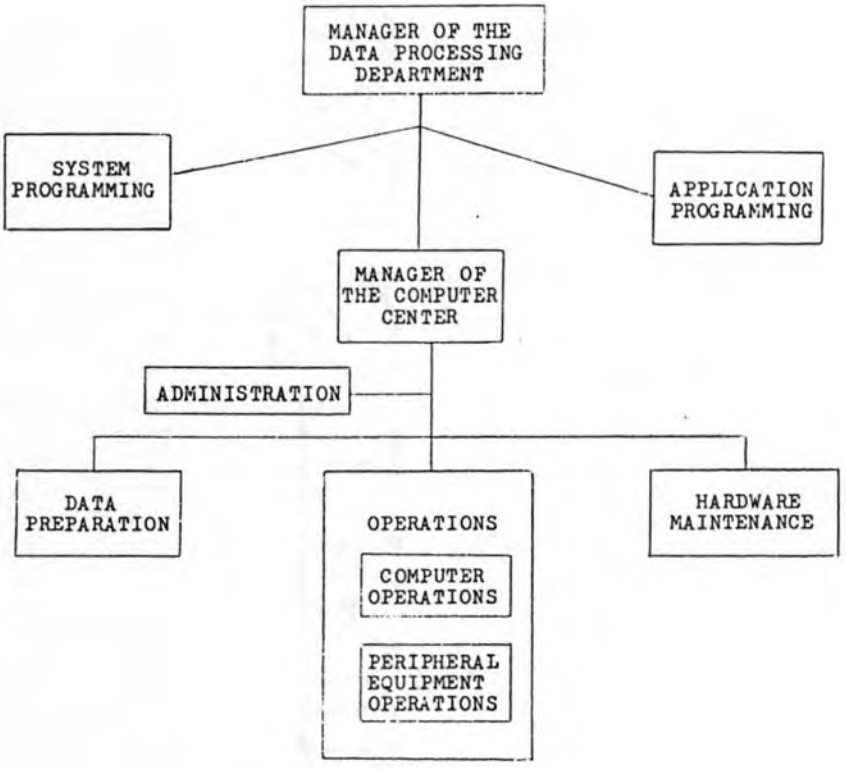


Figure 1.2. Categories of jobs in a typical computer center

3.1. Data Preparation and Hardware Maintenance

Data preparation involves DATA ENTRY EQUIPMENT OPERATORS who convert data from source documents into computer-readable form. In more advanced systems, users within the organization perform data entry without the aid of the operators.

The major part of hardware maintenance is completed by MANUFACTURER TECHNICAL STAFF in collaboration with the computer center staff.

Those functions offer only few interest for our purpose, they are, therefore, not considered furtherly.

3.2. Administration and Operating

The division of responsabilities between administration and operating is broadly influenced by historical features.

In the early days of computing, operators had no contact with the users and almost exclusively manipulated devices. The computing process was operator-limited; therefore, their work became quickly efficiency-driven. They knew many about devices but little about software and they relied upon the programming staff for all technical assistance. This image reaches the height of its fame with the sophisticated batch (including off-line operations,SPOOL...).

With time coming, systems became more and more complex. They are now integreted, parameterized (PRIOR), multimode (batch, time sharing, transaction processing) systems with important data management capabilities and designed to operate on a family of systems. This process influences the system management in two trends :

- On one hand, rightly or wrongly, the operators have kept their label of efficient devices manipulators and most of the newly appeared tasks were carried out by the administrator.
- On the other, the minute by minute control of the system itself has changed. It is became more complex involving much system messages and software knowledge than devices one.

These contradictory trends make the distinction between administration and operating today quite moving. The following note found in the "Bedienungsanleitung BS2000" manual [BEDAN75] illustrates this idea :

> " Die Aufgabenteilung zwischen Systemverwalter und Operateur ist nicht starr. Hier ist ein gewisser Spielraum in der Organisation des Rechenzentrums vorgesehen. In jedem Fall muB die Zusammearbeit zwischen Systemverwalter und

Operateur sehr eng sein. " (1)

However, the following will try to state a general repartition based on the operating session. Therefore, we define it first and we shall use it, as discriminatory element,to define roles inside the computer management.

3.3. Operating Session

The session is often related to one execution of the operating system program. For our purpose, we shall distinguish this session "sensu stricto" from the extended concept of operating session. In the following, when we use this extended concept, we shall specify it explicitly. To define it, let us consider the life of a given system occurrence as made of two steps : the system generation and its operating session. Figure 1.3 gives the states chart of a system occurrence life.

3.3.1. System Generation

The operating system is designed to run on any of a class of machines (7500 or 7700 series) at a variety of sites with a variety of peripheral configurations possibly changing in time. It must be configured for each specific computer site at the given time. Moreover, the system is not free from errors and, sometimes, it is modified to provide improvements.

The process which allows to manage this changes is known as the SYSTEM GENERATION (SYSGEN). To generate a system, a special program running on another one is used. It reads from a file the informations concerning the specific configuration and creates tables. It also selects modules from a precompiled library and links them together. The output, stored on a disk, forms a new SYSTEM OCCURRENCE ; versions provided by the manufacturer are only official stakes in those occurrences.

---

(1) The repartition of duties between system administrator and operator is not rigid. There is a certain freedom provided in the organization of the computer center. In any case, the collaboration between system administrator and the operator must be very strong.

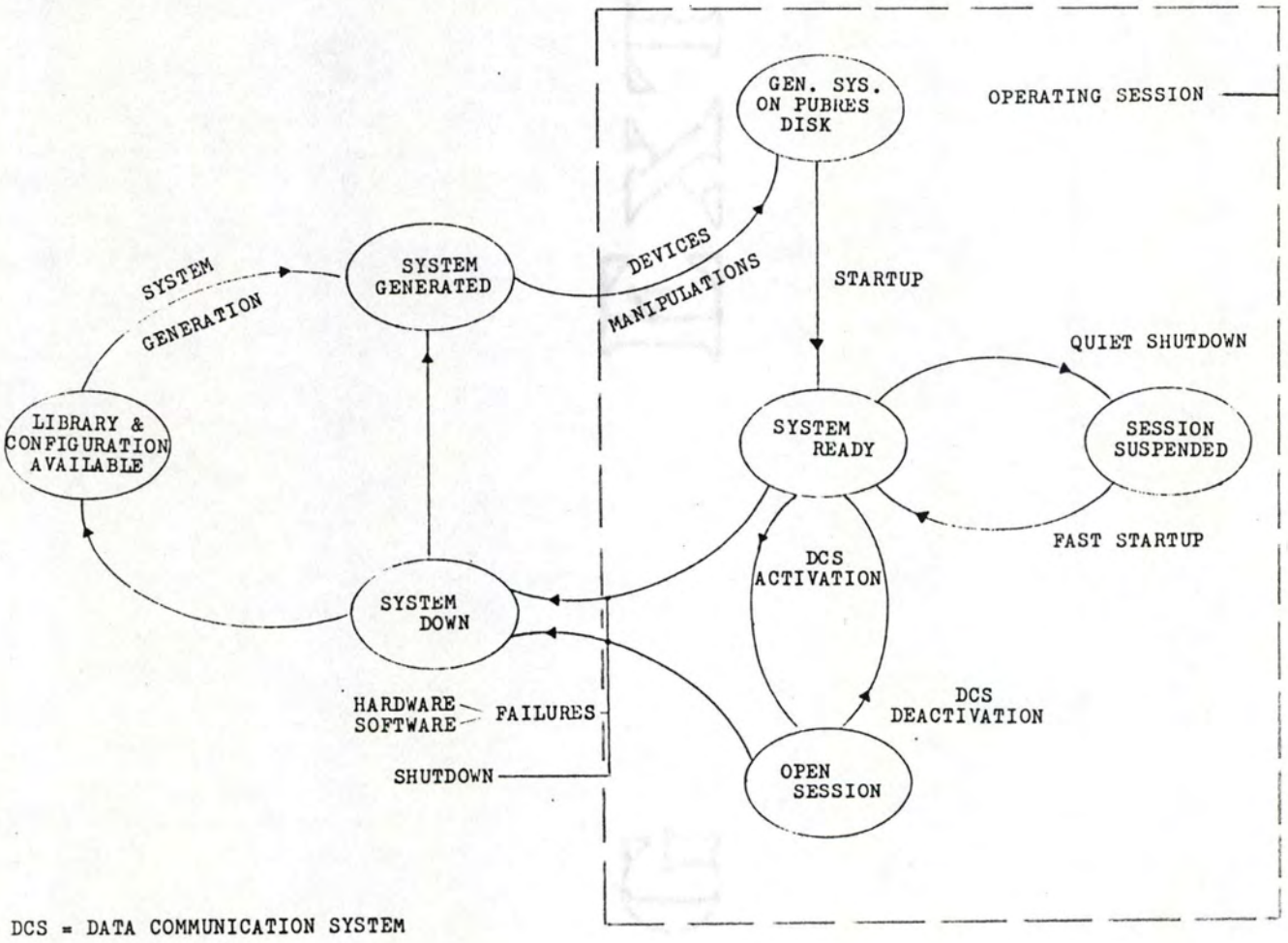


Figure 1.3. States chart of a system occurrence life.

3.3.2. Operating Session

The OPERATING SESSION is defined as the set of tasks associated with the life of a given system occurrence. The last one may die for many reasons, namely, hardware or software failures, configurations changes, policy changes, relieses from the manufacturer...

The operating session is just suspended if the system is stopped quietly, without modifications requirements. The reasons, generally external to the computer center, come namely from

- the rest of the organization: hollidays (nights, weekends), strikes...
- the computer manufacturer: hardware maintenance

3.3.3. The REPPING : pseudo-Generation

As manufacturers cannot afford to provide a new version of the system for each detected error, they provide a more economical mechanism consisting to replace directly pieces of the object code stored in the virtual memory. This correction operation, called repping (1), is completed during the STARTUP taking its inputs from so-called REP-files.

Direct object modification gives a new system occurrence without generation. therefore, one may consider it as a pseudo-generation, dangerous insofar as it creates a gulf between the situation "on paper" and the actually running system.

---

(1) This operation corresponds to the PATCHING in IBM terminology.

3.4. Administration

The SYSTEM ADMINISTRATOR (shortly administrator) is known by the system under the TSOS user identifier (USERID). His responsability consists to implement all the methods and tools necessary to the efficient working and the security of the system.

In practice, his role is made of a melting-pot of functions aggregated with the time. the principal ones are stated here after.

3.4.1. Users Administration

The adminstrator is the first and most privileged user of the system. As authorizer, he creates other users and controls their life (rights and priorities), assigns them projects and the associated rights and sets the accounting.

As "knower", he provides guidance to the end-users: documentation, informations about the current state of the system (hardware and software, particulary the utility programs as editors,compilers...) or error diagnosis help (dump analysis).

3.4.2. System Administration and Control

All system files are stored under the administrator's userid and he is responsible for providing utility programs to all other partners who can require them (in particular, the operators).

At the system control level, the operating session concept previously defined can be used. The scope of the administrator's responsabilities, in this field, is, indeed, everything OUTSIDE THE OPERATING SESSION ; this means

- SYSTEM GENERATION: in respect with the forecast load and the actual configuration.
- MIDDLE TERM POLICY: system and utility programs changes, performance studies.
- MAINTENANCE: failures diagnosis (dump analysis) and correction (reps).
- SYSTEM TUNING: playing on parameters (paging area size,priorities...)

3.4.3. Security

The security is a complex field which we do not intend to consider in details here. Briefly, the administrator is the AUTHORIZER at the users' level but his functions also involve the computer center. He generally plans the backup policies and supervises the operators for instance.

3.4.4. Computer Center Master

The previously defined functions are explicit, they are deduced from commands or service programs only available to the administrator. They also often place him at an informal key position:

- in the computer center: he acts as link between the users, the operators, the manufacturer's staff and the system.
- against operators: he often sets standards, monitors and evaluates their activities. Moreover, he is the following degree in the problem handling hierarchy, this gives him a skill power.

According to this key position, nearby but above the other partners, the administrator is often felt as the informal but actual computer center manager.

3.5. Operating

The scope of the operating is generally restricted to functions INSIDE THE OPERATING SESSION as previously defined. As any process, this session has an initialization (STARTUP) and a termination (SHUTDOWN). In between, the heard is shared by two parallel activities: those associated with the backup and those with the session "sensu stricto". The operating is structured in two parts.

3.5.1. Computer Operations

The COMPUTER OPERATORS (shortly operators) are in charge to complete STARTUP and SHUTDOWN operations. During the session, they monitor and control all operations, communicating with the system through consoles or local terminals.

Their responsabilities include pure manipulations aspects as well as some administrative power. Consequently, the distinction between the operators and the administrator's action possibilities inside the session is sometimes hard to be determined. The commands repartition between system partners shown at figure 1.4 illustrates this problem.

3.5.2. Manual Operations

The PERIPHERAL EQUIPMENT OPERATORS assist the computer operators. They insure an almost continuous computer processing by setting-up, operating and unloading peripheral devices.

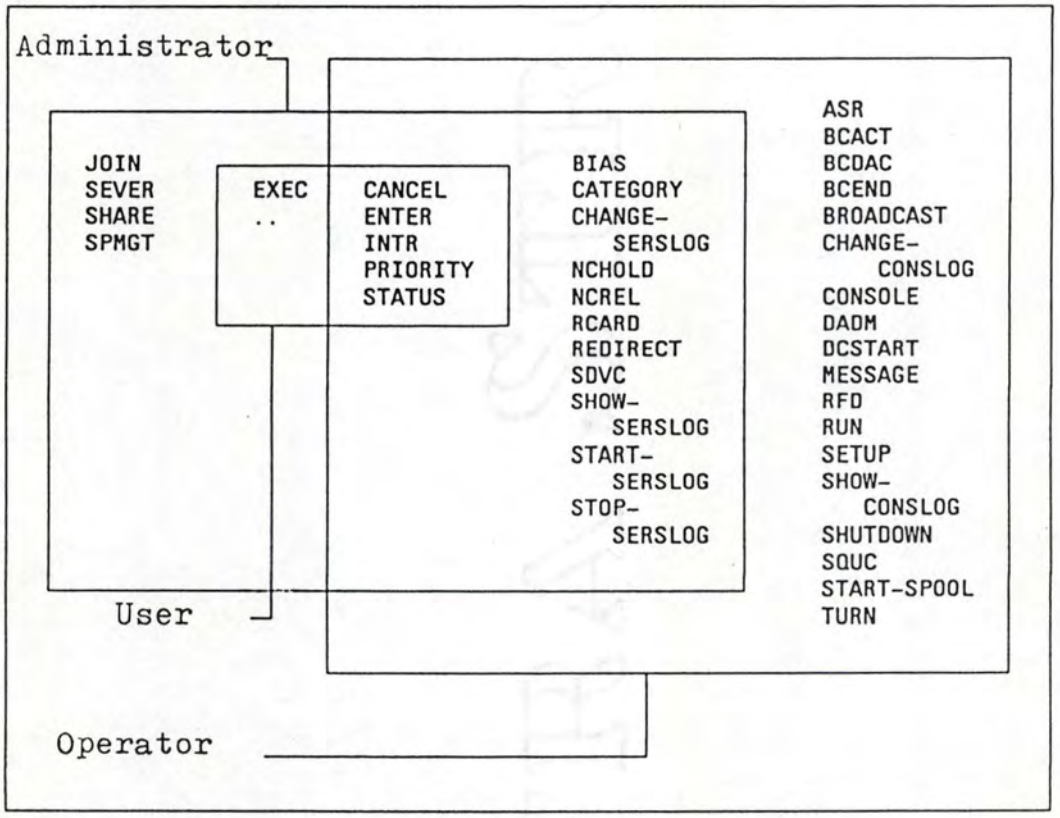


Figure 1.4. Commands repartition in BS2000

Chapter 2: Console Operations Concepts in BS2000

By CONSOLE OPERATIONS , one can understand the computer operations as previously defined unless the utility programs monitoring . Those utility programs are necessary to the good operating of the system, they involve mainly

- Logical and physical backup (ARCHIVE,FDDRL),
- Floppy or hard disk and tapes initialization (VOLINT,INIT),
- Memory dumps (SLED,SODA,SLME).

Some of them are SELF-LOADED (SLED,SLME). Once the system down (often after failures), they are loaded and monitored OFF-line from the console. Their monitoring is specific insofar as it is not controlled by the operating system. therefore, we do not consider it.

Some others run under BS2000 control but as the generic command allowing the execution of programs (EXEC) is only provided to users, they are not available from consoles. Therefore, one may not speak about console operations. The solution generally adopted to provide utility programs to operators consists to couple to the console a local terminal from which they work as user :

1. As ADMINISTRATOR : all programs are stored under the TSOS userid. Therefore the simplest solution is to make the operators work under this userid but it is very dangerous insofar as the operator holds all authorities controlled by the system.

2. As USER : the operator's power is reduced but still important. Moreover, the programs copies can be scattered.

In the following, the console operations will be defined as based on operating partners exchanging informations in the form of communications ; the communications distribution and the associated administration are processed by the console operations subsystem as shown in the figure 2.1. All transfered communications are strored in a special file for possible future access. This process is called logging.

The cited concepts will be studied in the following sequence :

1. Operating partners
2. Operating sources properties
3. Communications
4. Logging

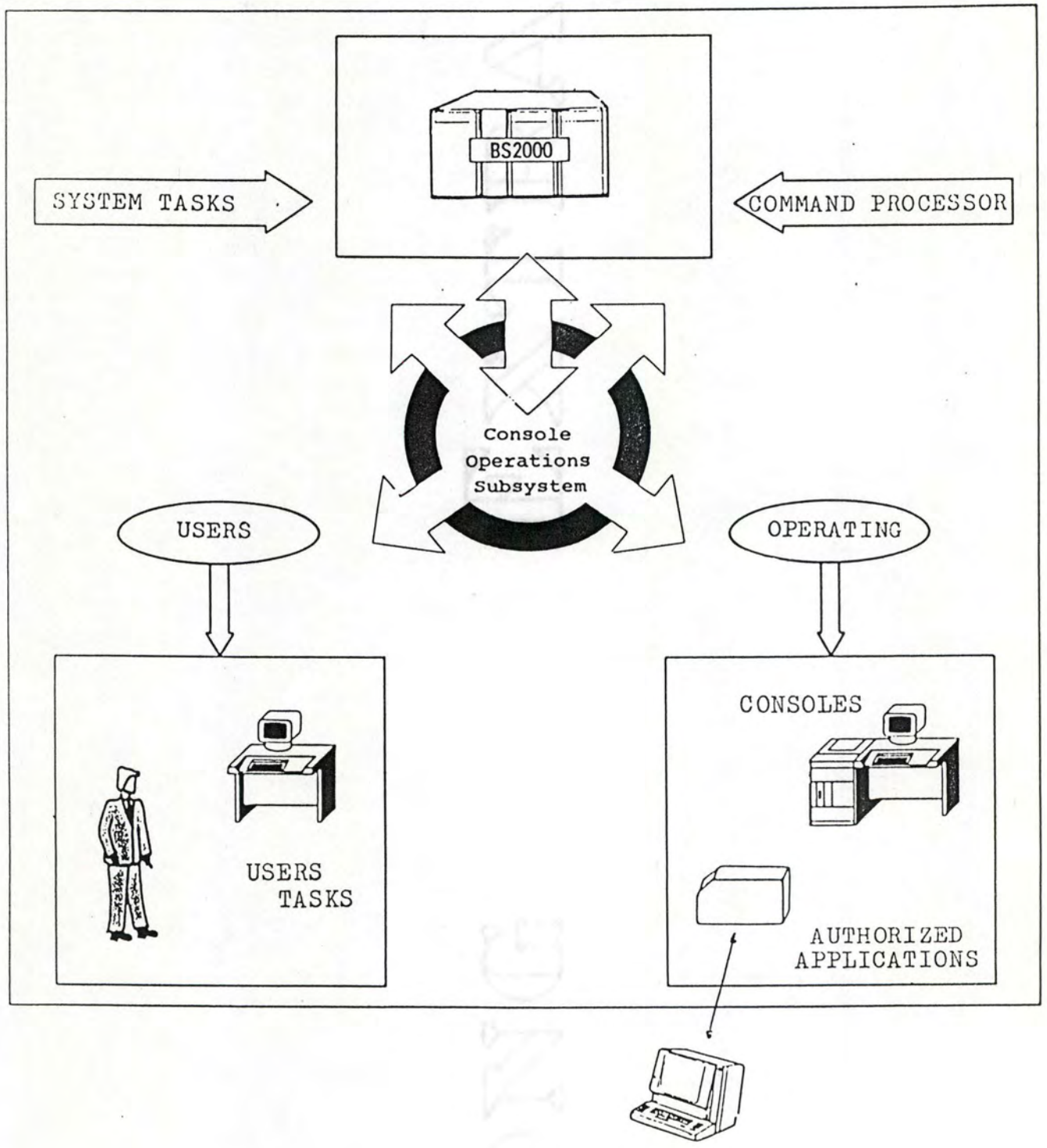


Figure 2.1. Console operations subsystem.

1. Operating Partners

1.1. User

A USER is either the administrator or any person or group allowed by him to connect himself (itself) to the system and to use it. A user is identified by a user identifier (USERID), he acts on the system through a task.

A USER TASK is a sequence of actions on the system required by a given user, it must begin with the / LOGON command. A user task is identified by a task sequence number (TSN). Among those tasks, only a limited number can be handled simultaneously; in this set, a task is identified by an internal task number (ITN).

1.2. Operating Sources

In the first computer generations, the operations involved mainly hardware manipulations and control completed from one physical console. With the time, as the systems complexity increases, their tuning and monitoring created new problems on top of hardware control. The number of messages reaching the console became then too big to allow one human operator to react to each of them quickly and correctly.

This bottleneck was suppressed by adding physical consoles and computer operators in order to reduce each of them own control area. The multiple consoles configuration introduced, however, many new difficulties. They were

- Economical : each console represents an heavy fixed cost
- Technical : competence areas definition, especially the consoles management and defective consoles handling

In the future, console operations automation seems to be a more economical and relatively secure solution to reach the system complexity. Multiple "consoles" configuration with cheeper consoles hardware (end-user terminals) is an other solution. In BS2000, a first step in those directions is completed by an interface allowing console operations from a special task which may eventually open a dialog with an end-user terminal (transaction processing).

The actual operating sources configuration integrates into its concepts these historical features. Their most important consequence is that the concept of operator is UNKNOWN by the system. Moreover, the recognized operating sources configuration is quite hybrid. One may, indeed, distinguish two completely different kinds of sources : consoles and authorized applications.

1.2.1. Console

Any local hardware configuration which can act as operating source is a console. Such a configuration can include screen, special keyboard, lights and switches panel, protocol printer, floppy disk deck, micro-processor (SVP)...

A console is identified by a MNEMONIC NAME as well as a CONSOLE NUMBER (theoretically < 24 but, in practice, < 4).

1.2.2. Authorized Application

An authorized application is a special task authorized by the administrator (at the generation) to complete some console operations. It is identified by an AUTHORIZATION NAME as well as by an APPLICATION NUMBER (< 64).

1.3. The System

The system software BS2000 is a complex structure; for our purpose, it is considered as made of SYSTEM TASKS identified by a TSN or an ITN as user ones. Some of those system tasks are activated automatically during the system initialization, they are called PRE-ALLOCATED TASKS and they have a pseudo-TSN made of letters.

2. Operating Sources Properties

2.1. Consoles

The consoles properties are related to the hardware oriented functions and the multiple consoles configuration handling.

2.1.1. Hardware oriented Functions

By hardware oriented functions, we mean essentially the system set-up and the CPU control. They are assumed from the so-called PHYSICAL MAIN or CENTRAL CONSOLE . Unfortunatly, those concepts mean different things; the following tries to clear them up.

The hardware oriented functions require some technical appliances, the global configurations containing them are marketted by the manufacturer under the label CENTRAL CONSOLE (3026-2..4 devices for instance). A central console can become defective; therefore, many computer center duplicate it. Only one of them, however, can actually assume the physical functions. This PHYSICAL MAIN CONSOLE is the set-up console. It remains on duty during the whole session.

Two more sophisticated tools are provided to the operator; the first, the SERVICE PROCESSOR (shortly SVP) is an independant micro processor which functions involve the CPU direct control (even off-line). This processor is often physically integrated in a central console so that one can name service processor, by language misuse, the whole console configuration.

The TELESERVICE , on the other hand, is a remote herdware control service provided by the manufacturer. His teleservice center is connected with the physical main console through a telephone link. The exchanged informations are stored, in the computer center, using a specific device called the TELESERVICE BUFFER . It is considered as a special console.

2.1.2. Multiple Consoles Configuration Handling

The multiple consoles configuration poses some new problems like competence distribution and consoles management or defective states handling. Let us have a look to how they are actually solved.

COMPETENCE DISTRIBUTION

The system is functionnaly divided in COMPETENCE FIELDS which ownership supposes the ability to

- receive messages related to the field defined by a so-called ROUTING CODE
- issue relevant commands defined by an ISSUE or AUTORIZATION CODE .

By now, authorization and routing are merged and identify a competence field as described in picture 2.2

COMPETENCE FIELDS	COMMANDS (Standard aniqnment)	AUTHORIZATION CODES
SYSTEM MONITORING	BIAS, CHANGE-CONSLOG, CATEGORY, EXCAT, IMCAT, MRSEND, MRSMOD, MRSSTA, MRSSTART, SHUTDOWN, STAM	R
DATA COMMUNICATION SYSTEM CONTROL	BCACT, BCASP, BCCONP, BCDAC, BCDISCON, BCDISP, BCEND, BCIN, BCLOSE, BCMOD, BCMOFF, BCMON, BCOUT, BCSWP, BCTIMES, DCSTART	C
JOBS CONTROL	ENTER, GETJV, NCHOLD, NCREL, SETJV	J
TASKS CONTROL	CANCEL, INTR, PRIORITY	P
DISKS OPERATING		D
TAPES OPERATING		T
PRINTERS AND CARDS PUNCHERS OPERATING	SQUC	O
CARDS READERS AND FLOPPY DISKS OPERATING	RCARD, RFD	I
DEVICES ADMINISTRATION	SETUP	G
SPOOL CONTROL	SDVC, START-SPOOL	S
REMOTE SPOOL MONITORING	REDIRECT	N
HARDWARE MAINTENANCE		H
SYSTEM ADMINISTRATION	CHANGE-SERSLOG, SHOW-SERSLOG, START-SERSLOG, STOP-SERSLOG	A
FILES ADMINISTRATION		U
GENERAL COMPETENCE	AGOGO, ASR, ASTOP, BROADCAST, MESSAGE, RUN, SHOW-CONSLOG, STATUS, TURN	E
MAIN CONSOLE FUNCTIONS	CONSOLE	*
MESSAGE PRINTED DURING THE STARTUP		V
BS1000-Simulation		F
FREE FIELDS		W, X, Y, Z

A COMPETENCE AREA is assigned to each console. Consisting of zero, one or more competence fields, it is identified by a set of routing codes.

The INTINSIC FUNCTIONS of a given console describe all functions (messages receipt and commands input) associated to its own competence area.

LOGICAL MAIN CONSOLE

The CONSOLES MANAGEMENT is a special competence field assigned to one and only one console at a time. This console is called the LOGICAL MAIN CONSOLE (shortly main console) (1). It should be identified by the "*" routing code (see figure 2.1). Unfortunatly, it is not always the case in practice.

The consoles management, coupled with two other functions, forms the main console functions, namely

- CONSOLES MANAGEMENT: to assign competence field to consoles, to switch them on or off.
- GARBAGE COLLECTION: to receive all relevant messages that cannot be distributed to other operating sources.
- EMERGENCY HANDLING: to receive the emergency messages and assume special functions necessary to prevent incoherent states (e.g. when the competence areas assigned to all consoles are empty).

REPLACEMENT FUNCTIONS

The life of a console can be seen as a sequence of STATES . The consoles states can be altered at two levels; at the physical level, it can be attached or not and operable or not (defective). At the logical level, the operator can switch it ON or OFF .

According to an AND combination of those components, a given console can hold 8 possible states divided in two classes: the state ATTACHED & OPERABLE & SWITCHED ON is called AVAILABLE STATE , all others being not available ones. When a console becomes not available, the system reliability imposes its competence area to be transfered to an usable one. In oder to perform this automatically, the

---

(1) The logical main console has only few links with the physical one described at the 2.1.1. The physical main is fixed at the STARTUP, it is the set-up console while the logical main can change during the session (because of the / ASR MAIN or the / CONSOLE SWITCH,OFF commands processing or because of a defective state). The first logical main console is the physical main console.

operators are given the possibility to define for each console a SUBSTITUTE CONSOLE (ErsatzKonsole). This console takes all functions assumed by the not available one. These functions are called REPLACEMENT FUNCTIONS , they involve the intrinsic functions of the first console (possibly including the main console functions) as well as all the replacement functions perhaps already collected by it.

TELESERVICE BUFFER EXCEPTION

The teleservice buffer is a specific device dedicated to the remote service processor; therefore, it will never be used for the dialog between the operator and the system. Consequently, it can be neither main console nor substitute console (but it can have such a substitute console).

2.2. Authorized applications

The authorized aplication connects itself to the console operations subsystem, acts as operating source assuming some intinsic functions (see competence distribution 2.1.2) and disconnects itself. As its lifetime is much shorter than a console, it can assume neither main console nor replacement functions. Furthermore, an authorized application has no explicit substitute console (1).

(1) The main console and sometimes other "*" routing code consoles play implicitly this role.

3. Communications

As previously defined, a communication is a piece of information exchanged by operating partners through the console operations subsystem. BS2000 knows five kinds of communications (Meldungen).

3.1. Messages

3.1.1. Definitions

A MESSAGE is a string of characters sent by any given operating partner to one or more operating source(s). A message may be an INFORMATION MESSAGE or a QUESTION .

A question distinguishes itself from an information message insofar as the sender partner cares for the reply coming from one of the receiving operating sources.

An information message is identified by a "%" and a question by a "?" in the string.

3.1.2. Task particularities

System and user task are considered in the same way with regard to communications exchanges. Consequently, they share the following particularities.

OUTSTANDING MESSAGES COUNTER

In order to prevent system disturbances (memory space saturation) because of a task sending too many messages, an OUTSTANDING MESSAGES COUNTER associated to each task stores the number of messages which distribution was asked but not yet completed. If this counter reaches a fixed limit L (L=16 for common tasks,more for some privileged ones but never unlimited), the sending task is suspended (PENDED) waiting for the unblocking.

WAIT FOR REPLY

The question asking by a task is a pure synchronous phenomenon. The task is, indeed, suspended until the corresponding reply comes. Now, this reply often depends on a human factor; therefore,it may never come and the "Cinderella" task will sleep for a long time. To prevent such a situation, two appliances are provided to the operator :

- The PROTECTED LINES on the console screen where opened questions may be logged

- The / STATUS MSG COMMAND which allows him to know which questions are still outstanding and to which of them he may reply.

Unfortunatly, these mechanisms do not reduce completely the deadlock risks.

TASK TERMINATION

A task may be normally terminated only if the associated outstanding messages counter is equal to zero. Otherwise, the task is put IN PASSLOOP , that is to say that it remains in the loop until the condition (here, OMC=0) is reached.

A task may also be aborted using one of the available /CANCEL COMMANDS between

- the operator /CANCEL command with or without KILL parameter,
- the end-user /CANCEL command. For our purpose, it is equivalent to the operator's one without kill parameter.

If the outstanding messages counter is equal to zero, ANY /CANCEL command is processed. The task is aborted. Otherwise, a /CANCEL command without kill parameter is rejected and a message "MESSAGE OUTSTANDING" sent back to the issuer although a / CANCEL KILL command is processed, killing the task.

3.1.3. Destination

A message destination is identified by a mnemonic name an authorization name or a routing code . Mnemonic and authorization nanes identify a single destination operating source. As defined at the 2.1.2, a routing code identifies a competence field. Such fields may be assigned to one or more operating sources; therefore, a routing code specifies a set of destinations.

3.1.4. Content

Messages are divided in two categories: standard or system messages and non standard messages.

Standard or system messages are stored in several special worfiles, each under the control of a given system component or utility program. Those system modules send them during the session; therefore one may speak about SYSTEM MESSAGES .

The output,however, may be requested by any task (user as well as system) calling the appropriate macro ((S)MSG(7)) so that it becomes difficult to still consider them as

system messages and the name STANDARD MESSAGES seems to be more accurate insofar as they are built according to the same formation rules which divide its content in four parts:

1. MESSAGE KEY : specifies the involved component and identifies
2. MESSAGES ATTRIBUTES : used for system internal processing control
3. MESSAGE TEXT : gives the message meaning in a given human language and often contains variable parts called INSERTS . Their number, position and contents are not formally defined.
4. COMMENT TEXT

Non standard messages are made of free text.

3.2. Replies

Replies are divided in two categories: normal and special ones.

3.2.1. Normal Replies

A NORMAL REPLY (shortly reply) is an string of characters sent from an authorized operating source to a partner if it is related to a question sent by this given partner. A reply is identified by a "." in the string.

3.2.2. Special Replies

A SPECAL REPLY is a piece of information sent by the VOLUMES HANDLER to signal a private volume mountage completion. The private volumes handling involves two asynchronous events: the mountage completion and the volume explicit request from a task. As soon as the volumes handler knows about a correct volume mountage completion, it tries to signal this event to the requesting task. Two different situation may occur:

- If the volume mountage was asked to the operator (at the human level), it can preceed the actual task request. Then, the reply is an answer to a not yet asked question.
- If the volume is not mounted when the task needs it, the requesting task is suspended and the operator receive an explicit mountage request.

3.3. Commands

A COMMAND is a string of characters sent to a special task called command processor task. its source can be

- an authorized operating source,
- a COMMAND FILE (RUN file): in this case, commands are sent one after the other with, for each,the reflection of the original operating source in order to allow authorization checks.

Commands are identified by a "/" in the string, theyare divided in two categories: normal and special ones.

3.3.1. Normal Commands

NORMAL COMMANDS are standardly defined and provided b the manufacturer. Their command processor tasks are systems tasks linked at the generation and loaded with the system during the STARTUP. We call them NORMAL COMMAND PROCESSORS .

3.3.2. Special commands

SPECIAL COMMANDS are very computer site depending. If they are also defined by the administrator at the system generation, their processors, the SPECIAL COMMAND PROCESSORS are dynamically connected to the system.

3.4. Additionnal Terminology

In order to clarify the following descriptions, two new arbitrary concepts are added

1. CURRENT FLOW COMMUNICATION: all kinds of communications previously defined.
2. NORMAL FLOW COMMUNICATION: any current flow communication unless special replies.

3.5. Rejection communication

A REJECTION COMMUNICATION (REJ-Meldung) is a string of characters sent by the console operations subsystem to the sender partner of an incorrect current flow communication

3.6. Emergency Messages

An EMERGENCY MESSAGE is a special message (information or question) output to the main console (at a physical level) when the standard message interface cannot be used. For instance, in case of

- inexisting messages handler (STARTUP),
- internal messages handler failure,

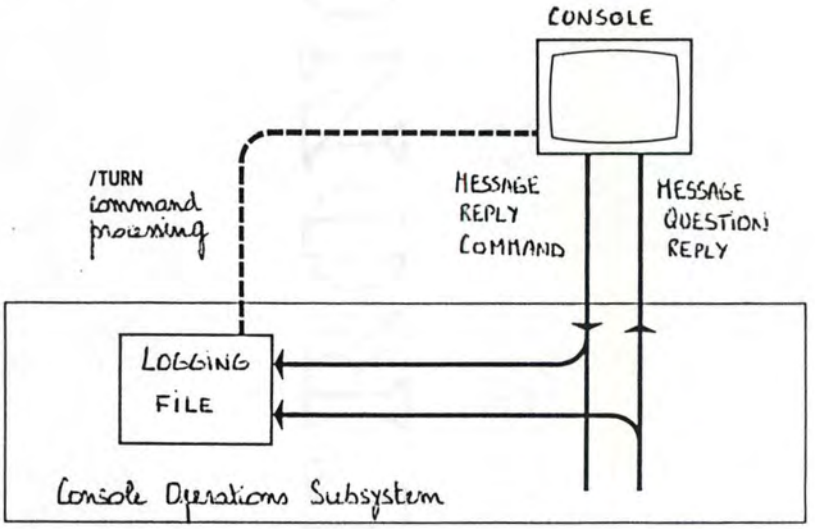
- critical resources failures (e.g. memory management),
- console queue jam...

4. Logging

All communications exchanged by the operators at the system consoles and their partners during the session, namely

- all current flow communications without taking possible syntax or semantic errors into account,
- rejection communication,
- emergency messages, if possible. (sometimes, the emergengy situation prevent it)

are logged in a special file named SYS.CONSLOG.ID-DATA. It can be accessed later using the /TURN command as desribed in figure 2.3.



Chapitre 3: Console Operations Subsystem : Processes Description

The CONSOLE OPERATIONS SUBSYSTEM is functionally divided in two parts: the most important one can be compared to a MAIL SYSTEM . It consists, indeed, to receive communications from several sources, to identify their nature, to check them and, finally, to distribute them to the right partner(s) and log them.

In front of these continuous functions, the console operations subsystem assumes, more contingently, some administrative duties. The actual operating sources hybridity separates them in

- REPLACE CONSOLE HANDLING and
- AUTHORIZED APLLICATIONS ADMINISTRATION .

In the following, the processes are analysed in more details. Our interest relies here on the functional aspects (independant from the actual implementation); therefore, we use the "Modèle de la Dynamique des Traitements" (Model of the Dynamics of Processing) proposed by F. Bodart [BODA83]. For more, see Appendix 1

1. Mail Processing

The communications are processed by the "mail system" in the form of a frame (information and control parts) called STORAGE QUEUE ENTRY (shortly SQE).

The processes chart presented at figure 3.1 shows how the mail functions are completed by four processes:

- NORMAL FLOW COMMUNICATION RECEIPT
- SPECIAL REPLY RECEIPT
- EMERGENCY MESSAGE HANDLING
- SQE DISTRIBUTIONS

Let us review them

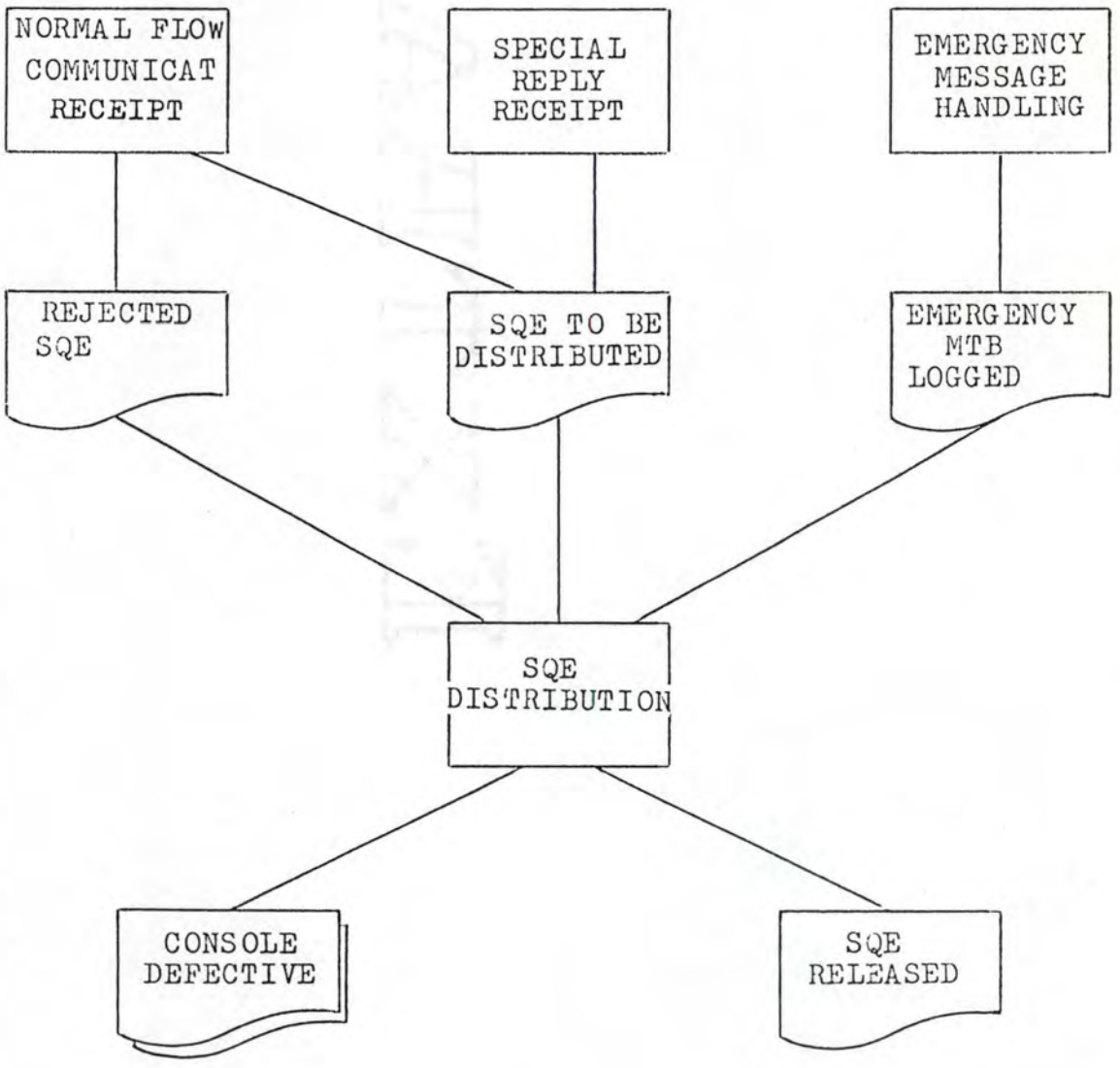


Figure 3.1. Mail processing.

1. COMMUNICATION INITIATION

This function creates the SQE and determines the communication source. It belongs to the following possibilities:

- TASKS: users, systems or command processor tasks
- OPERATING SOURCES: console or authorized application

If the source is a task, the associated OUTSTANDING MESSAGES COUNTER is incremented by one. If this counter reaches the fixed limit L, the task is suspended (PENDED).

If the source is a physical console, a LOCK is activated so that no further I/O can be completed from this input console before its current input acknowledgement.

2. COMMAND CHECKS

Four questions are asked about the command candidate:

- Should it be a command? (partial syntax check)
- Does this command exist? (existence check)
- Is the issuer authorized to use it? (authorization check)
- Is the command processor available (availability check)

PARTIAL SYNTAX CHECK

The command syntax is the following one :

< command > ::= <bl> / <bl> < opcode > |
<bl> / <bl> < opcode > @ < operands list >.
@ = one and only one blank

If the check UNTIL BEFORE THE OPERANDS LIST is unsucceful, an error is signalled (TEXT NOT RECOGNIZED).

EXISTENCE CHECK

If the operation code does not correspond to a command name declared at the generation, the command is rejected (COMMAND NOT FOUND). This operation code can be abbreviated since there can not be ambiguities with other operator's command.

AUTHORIZATION CHECK

An authorized application may issue a command only if the command issue-code is assigned to it.

A console may issue it if

- the command issue-code is assigned to it

1.1. Normal Flow Communication Receipt

In figure 3.2, a processes chart is presented that illustrates this receipt. The numbers in this figure are keyed to the following descriptions.

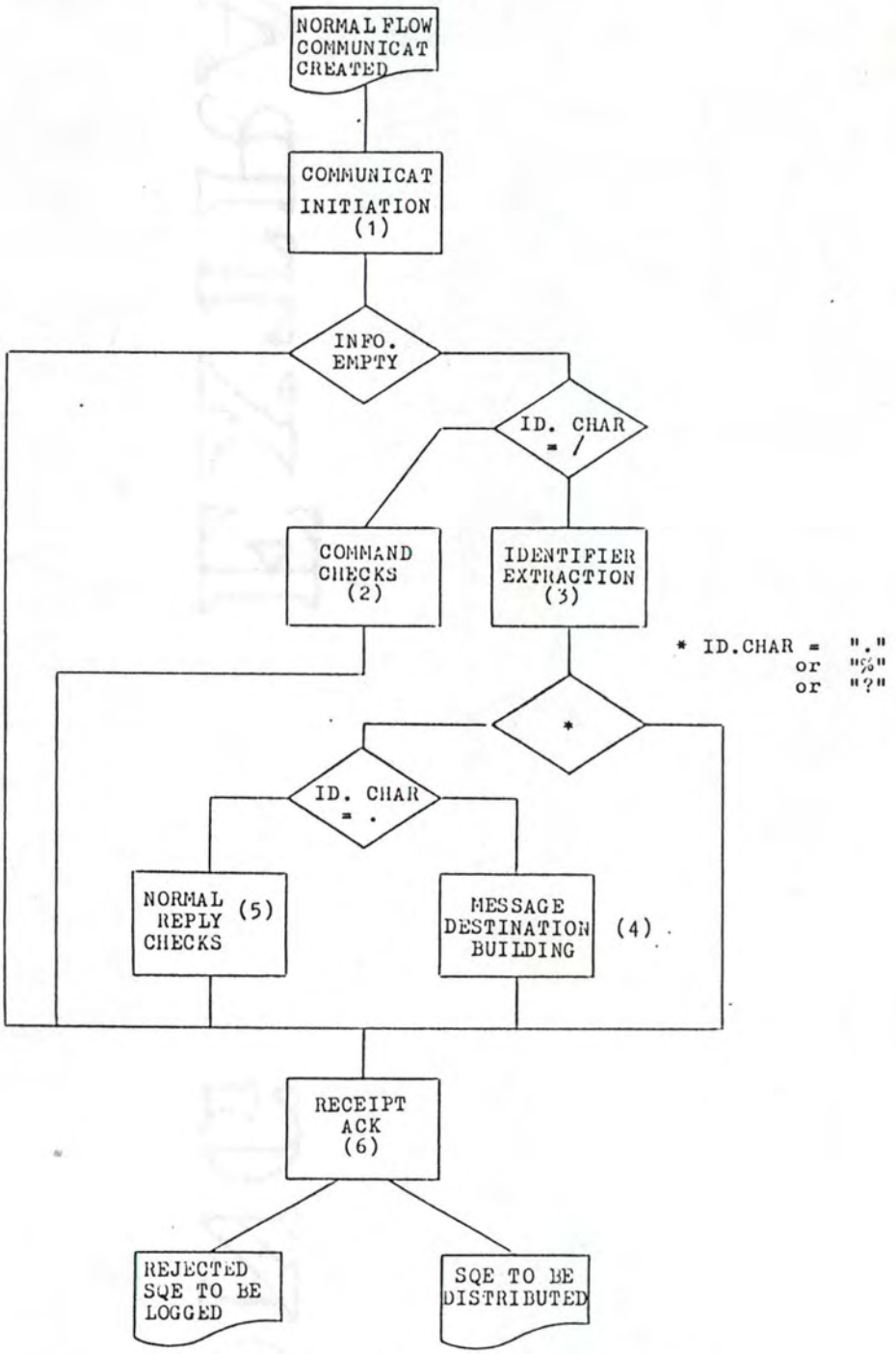


Figure 3.2. Normal flow communication receipt.

- it is the substitute console of a not available one to which the command issue-code is assigned.
- in version 8.0, it is the main console and no more physical consoles may issue it
- in version 8.5, it is the main console and the command is /ASR

If the check is unsuccessful, the command is rejected (COMMAND NOT ALLOWED) (1)

AVAILABILITY CHECK

If the command processor is not available, the command is rejected (COMMAND NOT AVAILABLE).

3. IDENTIFIER EXTRACTION

This function determines the nature of the non-command communication trying to find the identifier character. If it is impossible, an error is detected (TEXT NOT RECOGNIZED).

4. MESSAGE DESTINATION BUILDING

The destination identifier is a mnemonic name or an authorization name for a single destination and a routing code for a group.

FOR A ROUTING CODE

A routing code specifies a set of operating sources. Let us call S1 this original set. If the routing code is unknown by the system, a new set S2 is created using the "*" routing code. If this "*" routing code itself is unknown, a new set is created S3 = { main console }. The resulting set id divided in two separately handled subsets.

The original applications subset is modified according to the so-called filtering rules .

The original consoles subset (SS1) is modified first (in SS2) according to the filtering rules and, later, (in SS3) using the replace console handling informations .

---

(1) As exception to this rule, the command /CONSOLE and the commands /ASR DELETE, /ASR ADD or /ASR PRIMARY for other sources theoretically only allowed from the main console, are allowed here from all "*" routing code sources but rejected later by the command processor.

FOR A SINGLE DESTINATION

If the destination is incorrect, the error is detected (DESTINATION NOT FOUND).

Otherwise, the mnemonic name is used to build the destination taking a possible replace console handling into account. The authorization name identifies the actual destination application.

FILTERING RULES

Questions are always sent while the information messages sending is bound by two conditions:

- the operating source wishes to receive it, it can, indeed want to ignore ALL information messages (see /ASR command).
- its weight is greater than a level fixed by the destination operating source which can, indeed, ignore all not relevant messages (see /ASR command)

5. NORMAL REPLIES CHECKS

Four questions are asked about the reply candidate:

- Is it syntactically correct? (syntax check)
- Should it be a reply? (destination check)
- Does it reply to an outstanding question? (matching check)
- Is the sender authorized to reply? (authorization check)

SYNTAX CHECK

The reply syntax is the following one:

```
‹reply› ::= ‹BL› ‹commun data› "." ‹text› .
‹commun data› ::= ‹destination› ‹BL› |
                  ‹destination›  "-" ‹seq nber› ‹BL› |
                  ‹destination› "-" ‹P› ‹seq nber› ‹BL› .
‹destination› ::= "(" ‹MN› ")" |
                  ‹auth name›  |
                  ‹TSN›        |
                  ‹task letter› .
```

If no syntax error (TEXT NOT RECOGNIZED) is detected, the destination is checked.

DESTINATION CHECK

If the reply destination is a routing code, the reply is rejected (NO MATCH FOR REPLY). Otherwise, the matching is tried.

MATCHING

A reply has to be matched with an outstanding question, that means that a question is searched in the WAITING FOR REPLY QUEUE such as its destination is the origin of the reply. If the reply contains a sequence number, it must be correleted to the question one.

If the matching fails (NO MATCH FOR REPLY), the reply is rejected. Otherwise, the authority is checked.

AUTHORIZATION CHECK

User tasks are not allowed to reply. It may happen that non authorized applications manage to build a connection to the subsystem, their replies are rejected (USER NOT ALLOWED TO REPLY).

A console may reply if
- it is one of the question destinations
- it is the replace console of a no more available one which was allowed to reply
- it is the main console and no physical consoles can reply anymore because all authorized console are not available or all consoles have lost those competence (see /ASR command [BEDI75]) (1)

If the authority check is unsuccessful, the error is detected (USER NOT ALLOWED TO REPLY or TEMPORARELY ONLY MAIN CONSOLE ALLOWED TO REPLY). Otherwise, the reply processing is terminated.

6. RECEIPT ACKNOWLEDGEMENT

If an error was detected previously, a negative acknowledgement must be sent back to the issuer. Therefore, a rejection communication is created. Its format is REJ# ‹text› where the text belongs to those described between brackets here before. This rejection takes the place of the original communication as SQE TO BE DISTRIBUTED while this last one is time stamped to be logged.

---

(1) A console which has lost all its competence area can receive only messages which destination is its mnemonic name and cannot issue any command.

A time stamp is added to the SQE to be distributed. If the communication is

- a question: it is enqueued on the waiting for reply queue
- a reply: the corresponding question is dequeued from the waiting for reply queue (1)

1.2. Special Reply Receipt

As soon as the VOLUME HANDLER knows about a correct private volume mountage completion, it sents a response from task specifying the destination task, a return task and the replied text. Figure 3.3 illustrates this special reply receipt. The numbers in this figure are keyed to the following descriptions.

1. SPECIAL REPLY MATCHING
   The special reply matching is the same as the normal reply one described in this chapter at the 1.1 pt 6. If the volume mountage preceeds its request by a task, one can speak about a reply to a not yet asked question and, of course, no matching is possible.

2. SPECIAL REPLY SQE BUILDING
   The SQE is built and its destination determined.

3. SPECIAL REPLY REJECTION
   If the matching failed or no memory space is available, the return task is triggered. It will try to send the special reply again until success (question arrived or free memory space) or... shutdown.

---

(1) This implicitly implies that only the first reply to a question is taken into account.

1.3. Emergency Messages Processing

In emergency situation, a special system module (ECEMGMSG) takes a message out of a resident buffer and sends it to the console operations subsystem with the main console as destination.

If the main console is defective, the REPLACE CONSOLE HANDLING is triggered in order to find a new one and the output is retried. If an answer has been required, it is directly chained to the question.

The emergency situation can prevent the message logging (by resources lack or system failures); then, the message track is lost. Othewise, an EMERGENCY MESSAGE TO BE LOGGED is created with a "E" before the text part in order to allow further identification.

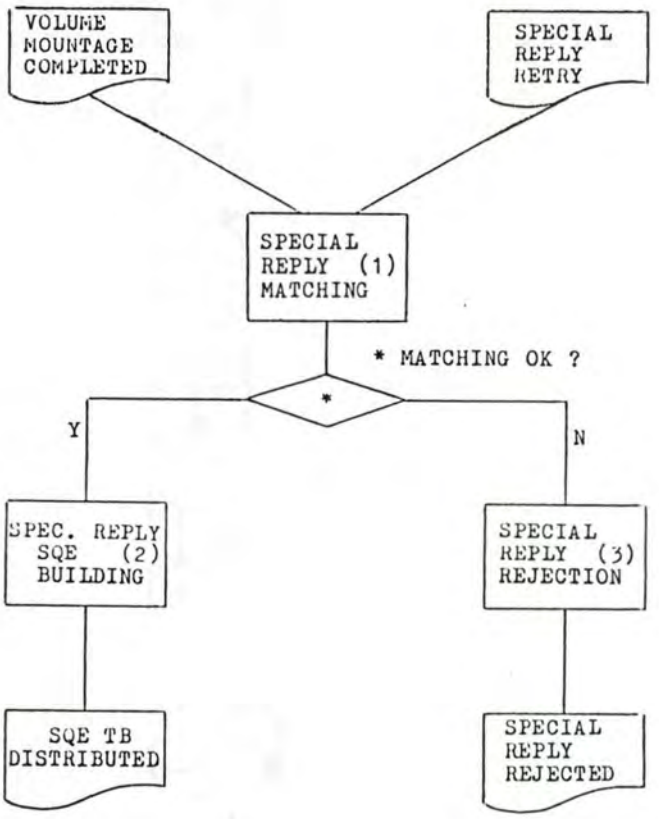


Figure 3.3. Special reply receipt.

1.4. SQE distribution

In figure 3.4, a processes chart illustrates the SQE-distributions. The numbers in this figure are keyed to the following descriptions.

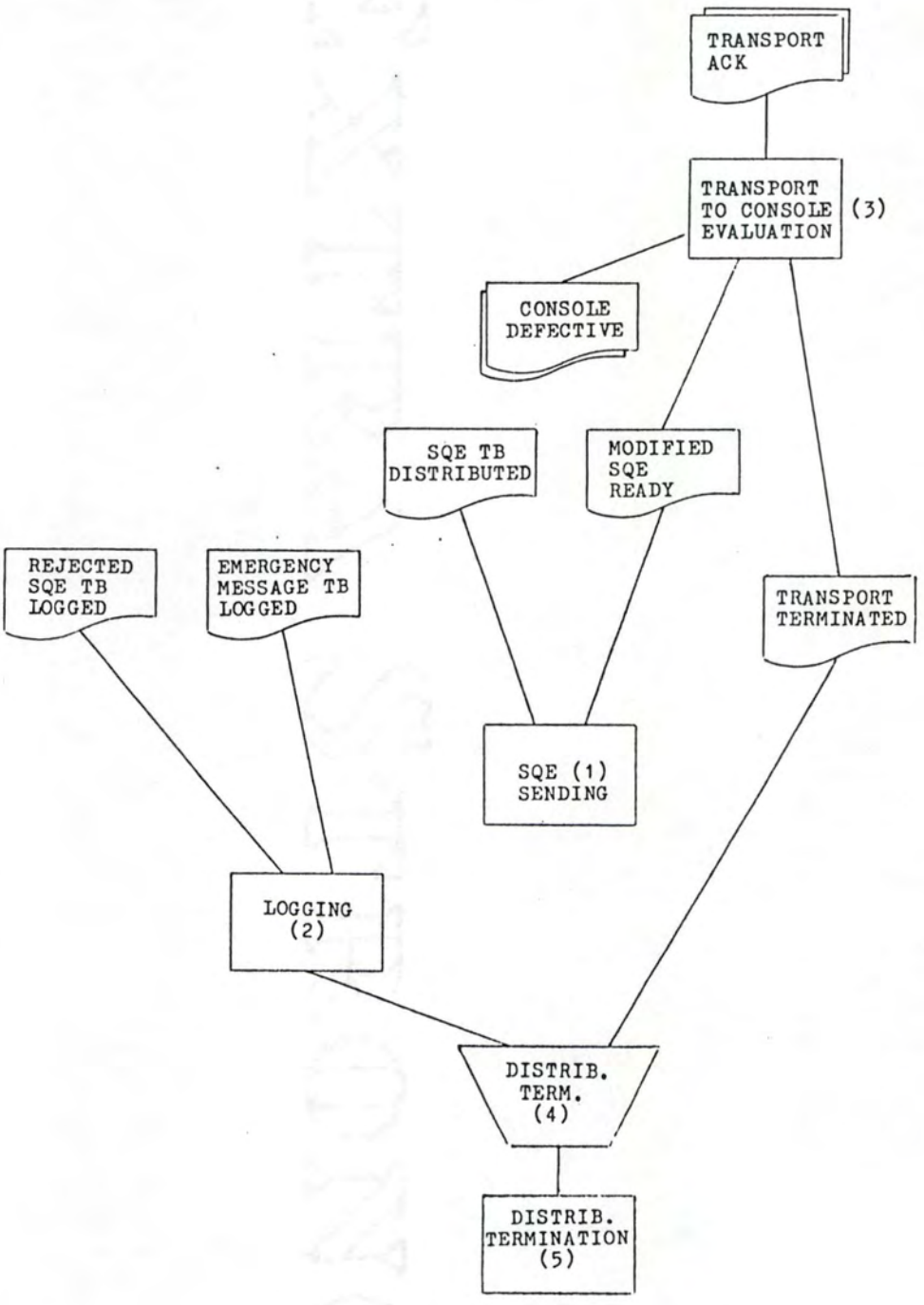


Figure 3.4. SQE distribution.

1. SQE SENDING

the SQE to be distributed is sent to the finally obtained destination which can be

- a single operating source (console or authorized application)
- an operating source(s) set
- a task (user, system, normal and special command processor tasks)

2. LOGGING

3. TRANSPORT TO CONSOLE EVALUATION

SQE transport must be acknowledged by all consoles but only by consoles. For each negative acknowledgement provided by a console, a CONSOLE DEFECTIVE event is created. To be successfully terminated, a SQE transport must reach some conditions:

- For a question, at least one positive ACK must come back from destination physical consoles
- For an information message, a positive ACK is necessary only if the console subset given by the filtering rules is not empty.

If the transport is not successful, a new destination is created with the main console as original set and according to the replace console handling informations and the transport is retried. Otherwise, the transport is terminated

4. DISTRIBUTION TERMINATED

The synchronization point is completed if the logging is terminated in case of emergency message or rejected communication and if the logging and the transport are terminated for all other SQEs.

5. DISTRIBUTION TERMINATION

If the source was a task, its associated OUTSTANDING MESSAGES COUNTER must be decrementd by one. Finally, the SQE must be destroyed.

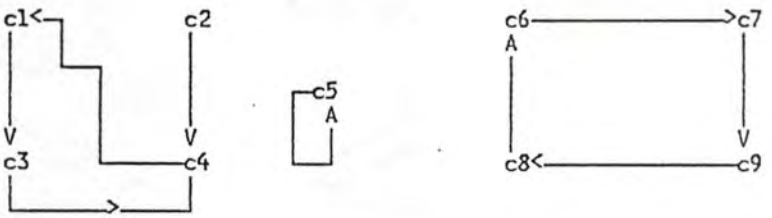
2. Replace Console Handling

Supplied only for physical consoles, it involves
- replace console assignment
- replace console handling
- replace console informations access

2.1. Replace console assignment

As defined previously, operators are given the possibility to define for each console a REPLACE or SUBSTITUTE CONSOLE (a console being allowed to be its own replace console) used when the primary one is no more available. The substitute console are defned at the system generation either by the administrator or, if he does not make it, by an internal algorithm. They may also be modified during the session from the main console using the /CONSOLE DEFINE command.

One can represent the current replacement situation by a graph
G = {C,R} where
C = {consoles}
R = { ri = (C1,C2) such as
C1 E C,
C2 E C
and C2 is the replace console of C1.



A CONSOLE REPLACE PATH is the set of console(s) possibly connected using ri relations. Such a path contains always a cycle (which makes the mapping easier).

2.2. Replace Console Handling

The replace console handling is triggered by consoles state changes from or to the available state. As these changes can be combined, an already inoperable declared console can be switched off and the replace console handling triggered twice.

2.2.1. Available to not available

A console can be declared defective by a console defective event and switched off during the /CONSOLE SWITCH,OFF command processing.

2.2.1.1. Console defective event

Its source may be an I/O failure detection during the SQE transport or a busy state greater than three minutes. Each 20 seconds interval, indeed, a special timer interrupt occurs, the consoles state is checked. If a console is 10 consecutive times in busy state, it is declared defective.

The ACTUAL REPLACE CONSOLE SEARCH is triggered. If a new main console had to be found, the message "E902 MAIN CONSOLE INOPERABLE. REPLACED BY THIS ONE" is sent to all "*" operating sources. That means that several sources can think they are the new main console (even authorized applications!) but,of course, there is only one main console. If a new auxiliary console has been found, the message "E903 CONSOLE ‹Ki› INOPERABLE" is sent to all "*" routing code operating sources but no explicit message gives the actual replace console of the defective one.

2.2.1.2. Console switch off

The available consoles number has to be at least one, if it should no more be the case after the /CONSOLE SWITCH,OFF command processing, it is rejected with the message "E650 THIS CONSOLE IS THE LAST OPERABLE CONSOLE IN THE SYSTEM. /CONSOLE CMD REJECTED".

Since version 8.5, the main console is no more allowed to be switched off so that the previous condition is secured and only the second step of the ACTUAL REPLACE CONSOLE SEARCH is necessary. The message "E652 CONSOLE ‹Ki› GETS ALL FUNCTIONS OF CONSOLE ‹Kj›" is sent to the actual replace console.

2.2.1.3. Actual replace Console Search

It is made in two steps:

1. Trial to find an available main console organized in emergency levels:
   - the main console is available
   - scan of the main console replace console path
   - scan of other replace console paths
   - forced by system switch on of an operable switched off console
   - pool of all consoles in order to take the first operable again one as main console

2. trial to find a replace console for the auxiliary one if necessary:
   - scan of the auxiliary console replace console path
   - take the main console as replace one

2.2.2. Not available to available

The available again console takes back its intrinsic and possible replacement functions but the main console functions are not restituted .

2.2.2.1. Defective to operable again

This may be triggered, implicitly, by an input arrival from a formerly defective console or, explicitly, signalling that the console is operable again.

2.2.2.2. Switch off to on

The /CONSOLE SWITCH,ON command issued from the main console can involve a console either operable or not. It is normally not possible to check it reading in tables; then, it is useful to perform a physical test I/O in order to consder as available again only the operable consoles.

2.2.3. Main Console special Features

The only way to replace the available main console by an other one (since the main console functions are not re-set at the come-back from not available state) consists in issuing the /ASR MAIN command from an auxiliary console. This command processing is bound to the main console operator agreement.

If he agrees, the source console takes the main console and the "*" issue code (if it did not have them earlier) functions. The old main loses its main console functions and

the "*" routing code one unless it had them before becoming main. Otherwise, the command is rejected.

2.3. Replace Console Information Access

The /CONSOLE HELP command allow a console to know which console is its actual (perhaps not the "on paper" defined one) replace console.

3. Authorized Applications Administration

The console operations subsystem manages the authorized applications connections and diconnections. During the connection handshaking, the patner's nature and its authority must be checked. A connection break must be recorded. Those functions belong to a more general set of administrative functions highly depending on the physical architecture. We depict them in the next chapter.

Chapter 4. Console Operations Subsystem :

ARCHITECTURE

A general description of the physical architecture is given at the figure 4.1. The UCON-task (universal console task) can be considered as the CENTRAL NODE of the console operations subsystem; the main function of its module NBROUTE consists, indeed, to complete the functionalities formerly defined. In addition to these mail system logical functions, NBROUTE assumes some more "sources oriented" ones:

- physical console I/O handling
- DCAM-applications handling

The single input interface of NBROUTE is a two chambers bourse (1) called ROUTING BOURSE . Events pended on this bourse signal communications coming from

1. A TASK

   All tasks, users, system or normal command processor tasks are considered in the same way. The module ECTYP running under their TSN, controls the message, builds an SQE, transmits its address to NBROUTE and pends the task.

   If the communication source is the NKV-task (from the volume handler NDM), NBROUTE has to build the SQE.

2. A DCAM APPLICATION

   Authorized applications and special command processors are DCAM applications. They communicate with the system with letters NBROUTE receives the letter announcement and builds the SQE.

3. A CONSOLE

   A console I/O involves two partners. NBROUTE is the logical partner, while the physical one is made of two parts : the UCON-task module NBCONS and the KTT-task (console driver task) which assumes screen control, device failure handling, physical I/O start. NBROUTE receives the message from the consoles and builds the SQE.

---

(1) BOURSES are a special synchronization appliance provided in BS2000. For more, see Appendix 2

The SQEs are stored in class 3 memory (resident and dynamically allocated) due to physical I/O features. Their length may be at most 255 bytes in which 52 bytes are reserved for administration so that the real information part length may be at most round 200 bytes.

The physical architecture is articulated upon four directions:

- NORMAL COMMANDS PROCESSING
- PHYSICAL CONSOLE HANDLING
- DCAM APPLICATIONS HANDLING
- LOGGING

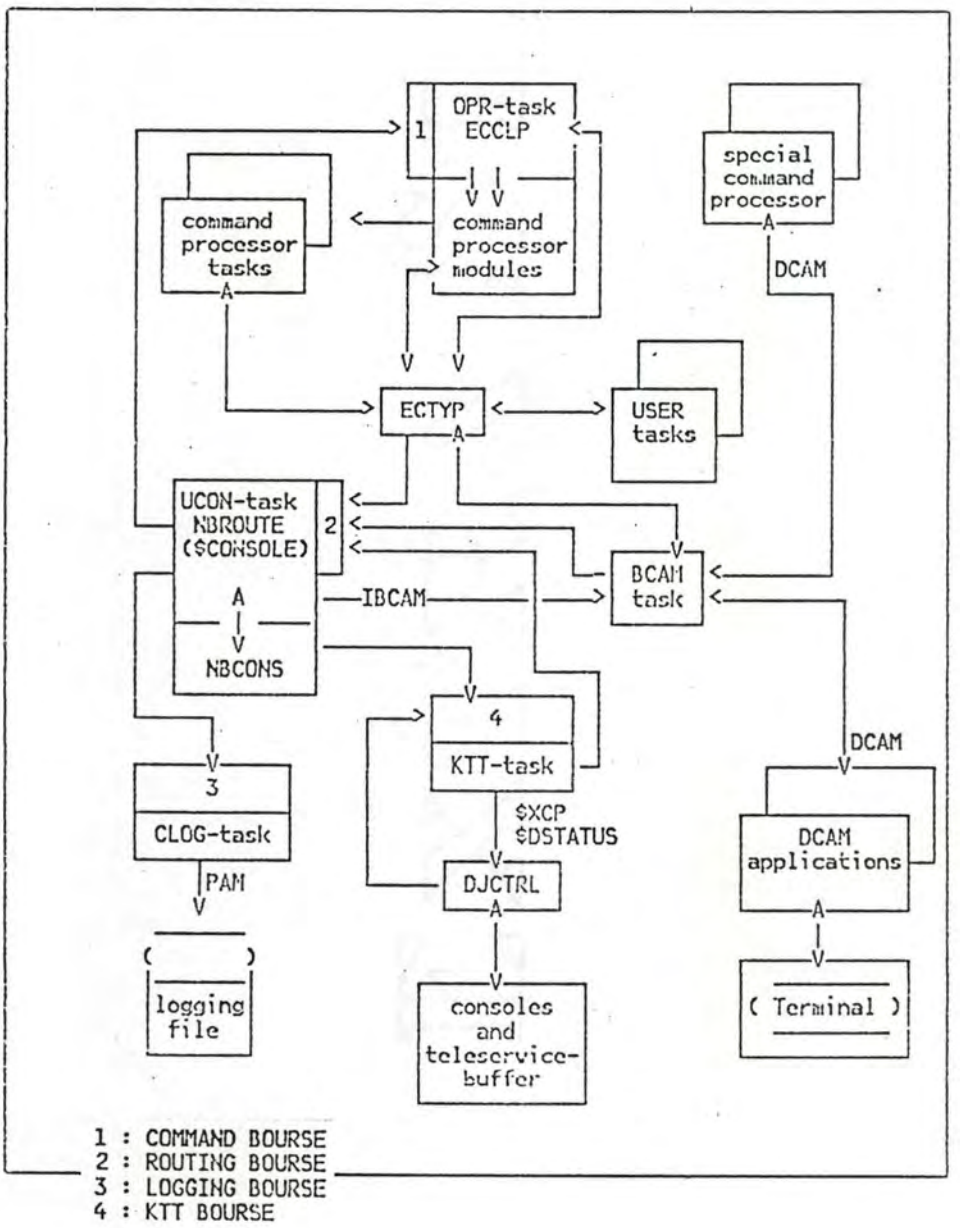


Figure 4.1. General Architecture

1. Normal Commands Processing

NBROUTE checks the command as previously described and sends it to the OPR-task (operator task) using the COMMAND BOURSE interface.

The OPR-task main module ECCLP
- receives the SQE address and stores it in a standard NBROUTE pointer (ADOPCMD2) so that NBROUTE keeps the SQE track for possible further handling. As command are processed one after the other, there is no crash risk.
- prepares the syntax checker (ISP) parameters list
- activates the appropriate processor module (using BALR and taking the programming language, assembler or SPL, into account) or processes itself the command (e.g. the /TURN command [BEDI75]).

The OPR-task processor module (ECCLP or an other one)
- checks the operands syntax
- processes itself the command possibly exchanging informations with the command issuer using the ECTYP interface. The message destination may be a routing code (by rule only for the OPR-task but...); then, NBROUTE builds the correct destination accessingthe command SQE (using ADOPCMD2).

In some cases, it should be dangerous (DEADLOCK) to use ECTYP interface from the OPR-task; then, the OPR-task processor module creates an auxiliary task ($CREA), triggers it, gives it the necessary informations and gives the control back to ECCLP as if the processing was completed.

In other cases, the command processing is to be completed by an already created and triggered task (UCON,BCAM..); then, the OPR-module sends it the necessary informations and gives the control back to ECCLP as if the processing was completed.

As the OPR-task cannot transfer the original SQE to an other task (paging error), it gives them copies (generally data oriented parameters lists).

In some last cases, the command must be rejected while correct because of an other instance of that command is in progress and two parallel processings would threaten the system consistence. It is the case with the /TURN, /ASR MAIN and /CHANGE-CONSLOG commands.

When the processing is completed or assumed as such and the still outstanding messages distributed or rejected, ECCLP tries to release the SQE (if the logging is done).

2. Physical Console Handling

2.1. Consoles Input and Output

Each I/O is based on a chain structure between the two partners described in the intoduction of this chapter. The communication from the physical to the logical partner uses the current UCON input interface: the routing bourse. The communication in the other direction involves the triggering of NBCONS using the BALR interface so that, normally, NBROUTE does not have to communicate directly with the KTT-task.

As consoles are protected by a hardware lock, the I/Os are completed one after the other and only one input buffer is necessary for normal communications. As output has many sources and only few destination consoles, an output queue is necessary. In version 8.5, this logical queue is divided in three parts:

- the OUTPUT QUEUE controlled by NBCONS
- the OUTPUT ORDERS from NBCONS pended on the KTT-BOURSE
- the SCREEN IMAGE BUFFER handled by the KTT-task

2.1.1. Input from Console

An input from console causes an interrupt called "ATTENTION INTERRUPT". Its exact source (AUFRUF key...) depends on the hardware interface. The interrupt analyser (P3) triggers (BALR) the DJCTRL that pends an AI order on the KTT- bourse.

The KTT-task, reaching the AI order in its bourse, first activate a hardware lock on the input console so that no further I/O can be completed before the input acknowledgement from NBROUTE. After this, it starts the physical I/O with the appropriate CCWs and the $XCP macro's depending on the hardware interface (CFCF-KCON,CFCF-DCON,CFCS). The physical I/O may be a complex sequence of physical actions (one CCW processing); during their completion, some events (hardware failures, CANCEL key pressure, time out, no ETX) may cause input abort with return message in format " *** ‹text› ". For easiness, let us consider the physical I/O as an ATOMIC action.

After I/O completion, the DJCTRL is triggered by an other interrupt called "TERMINATION INTERRUPT" and sends a TI order to the KTT-bourse.

The KTT-task reaching this order, analyses the I/O result (device status: $DSTATUS). If it detects a failure, it sends a "CONSOLE DEFECTIVE" return code to UCON, clears the defective console SCREEN IMAGE BUFFER and sends a negative acknowledgement "I/O UNSUCCESSFUL" for each SQE of this buffer. Finally, it resets the harware lock.

Later, it will send a negative acknowledgement for all messages output orders involving this console which were pending on the KTT-bourse at the defective state definition as well as for all SQE in the same situation in the NBCONS output queue.

If the input was correct, the KTT-task sends an "INPUT ARRIVED" event to UCON.

NBROUTE, reaching the "INPUT ARRIVED" event, builds an SQE, checks it as previously defined. If it accepts the input, the acknowledgement, an "ACTION FREE" order is sent back to NBCONS. If it reject it the negative acknowledgement is coupled to the rejection communication.

The acknowledgement, after transit by NBCONS, reaches the KTT-task which resets the hardware lock on the console and sends the possible rejection communication.

Unfortunalely, the chain structure is only partly respected for commands input, the free acting of the input console happens, indeed, BEFORE the command processing completion so that two commands can be in progress at the same time; furthermore, the second one processing can pass beyond the first one and the commands processing communications can then be mixed. this, of course does not simplify the operator's work.

Such command processing communication does not contain acknowledgement, they are

- a command check result (from OPR-task or a command processor module)
- a question (from a command processing task)
- a command processing result

sent using the normal interface ECTYP without REJ-format.

In version 8.5, no more distinction is made at the physical level between ON and OFF console (the switch is exclusively logical) so that input from a switched off console may reach the UCON-task and NBROUTE has to reject the input with an explicit rejection communication.

2.1.2. Output to console

NBROUTE builds a distribution mask (implementation of the set described earlier in chapter 3). For each pointed console, NBROUTE calls NBCONS with the order "SEND TO CONSOLE".

NBCONS pends the SQE on the OUTPUT QUEUE and

reverberates each output order on the KTT-bourse.

The KTT-task, reaching an "OUTPUT TO CONSOLE" order, tries to store the output in the SCREEN IMAGE BUFFER. If this last one is full, it sends the appropriate return code to the ROUTING-bourse for NBCONS. Depending on the screen update mode (time interval, roll-up or manual), it waits or not for output triggering events. When they occur (if necessary), it sets the hardware lock and starts the physical I/O giving orders to the DJCTRL. Once again, let us consider this physical I/O as an atomic action.

The "TERMINATION INTERRUPT" handled by the DJCTRL, reaches the KTT-task which analyses the I/O result (device status). If it detects an error, it reacts as in the case of an input error. Otherwise, it sends a positive acknowledgement "I/O SUCCESSFUL" for the invoved SQE to UCON and resets the hardware lock.

In case of /CONSOLE SWITCH ,OFF command processed by NBROUTE, it may happen that messages previously sent have not yet been physically distributed. To prevent messages loss (or at least long delay), NBROUTE has to ask explicitly to NBCONS to send it back the logically lost messages. To do so, it calls NBCONS with action "CLEAR CONSOLE QUEUE" and NBCONS reverbate the order to the KTT-task.

2.1.3. Attach and Detach

Formerly, the console physical state involved the defective/operable component but also the attached/detached one. In version 8.5, the scope of these concepts is reduced. The Nucleus Device Management (NDM), processing the /DETACH or /ATTACH devices configuration modification commands, signals "CONSOLE ATTACHED" OR "CONSOLE DETACHED" event to the KTT-bourse. KTT simulates then "CONSOLE DEFECTIVE" or "CONSOLE OPERABLE AGAIN" events to the ROUTING-bourse so that NBROUTE does not care anymore for those physical states changes.

3. DCAM Applications Handling

3.1. General Administration

3.1.1. $CONSOLE Application handling

$CONSOLE is a

- SYSTEM (then it begins with $)
- BCAM APPLICATION: a BCAM application may be seen as a mail boxes pool by which a task or a group of tasks (often also called application by language misuse) makes it know by the Data Communication System (1). Its creation called OPENING is completed by the PRIMARY TASK , linking other secondary tasks is called ENABLING of the application.
- PREDEFINED DURING BCAM RESOURCES DEFINITION: meaning that
  1. the application is implicitly opened by the BCAM-task that will announce events concerning it even if it is not yet enabled.
  2. An authorization key can be specified in order to protect the application against not authorized enabling.
  3. The application is not bounded by limitations which are defined for current applications

3.1.1.1. BCAM ready Handling

As soon as BCAM is available after the /DCSTRART command processing, it signals a "BCAM READY" event to NBROUTE which opens an application with the following characteristics:

- APPLICATION NAME = $CONSOLE
- TRANSPORT ACNOWLEDGEMENT = NO
  A letter sent is bounded to an immediate BCAM access check and, after assumed to be correctly transfered.
- GRADE OF SERVICE = BASIC
  The set of transport services and functions available for future connections is basic.

---

(1) The Siemens Data communication System is called TRANSDATA. For more, see Appendix 3.

Particularly, the maximal letter length depends on this grade of service.

- QUALITY OF SERVICE = NORMAL
  The recovery from data transfer related errors is not provided.

3.1.1.2. BCAM Shutdown Handling

The BCAM shutdown because of failures or /BCEND command processing causes a "BCAM SHUTDOWN" event to be announced to NBROUTE. NBROUTE breaks logically ALL connections, closes the $CONSOLE application ($DESABAP) and sends an error message to the console.

3.1.2. Connections to $CONSOLE

A CONNECTION is a logical path between two TRANSDATA transport partners. One of these partners must be an application, the other can be a terminal, an application or any logical equivalent (e.g. in BS1000).

The connection building is bound to the following protocol: the calling partner requests a connection. It sends a REQCON order and a connection letter that reach (or not) the other partner. The called partner receiving a connection request can accept or reject it. The possible connection confirm is also accompagned by a connection letter. In case of called partner silence after a given timeout interval, BCAM rejects the connection. The caller receive then the acknowledgement for its connection request.

After this handshakking, the partners exchange letters or telegrams until the connection break. This connection break can be implicit (BCAM shutdown, application disabling) or explicit (the connection close is unilateral).

3.1.2.1. Connection Request

A "CONNECTION REQUEST" event sent by BCAM to UCON triggers NBROUTE that receives the CONNECTION LETTER ($RECLET) and checks

1. PARTNER NATURE: it must be an application. Terminals can be used for operating with an application as interface but a terminal is not an authorized partner.

2. CONNECTION LETTER SYNTAX: using ISP syntax machine with the encryptor if necessary (encryptted connection password).

3. PARTNER AUTHORITY: connections to $CONSOLE are only allowed to applications which names were defined at the system generation.

If the checks are successful, NBROUTE accepts the connection. It sends a welcome letter, records the connection and sends the appropriate message to the consoles. Otherwise, it rejects the connection request without message sending to the consoles.

3.1.2.2. Connection Break

Reaching a "CONNECTION BREAK" event coming from BCAM, NBROUTE records it without message sending to the consoles.

3.2. Letters Handling

The data exchanged by TRANSDATA partners in a connection can be organized in LETTERS or in TELEGRAMS ;the $CONSOLE interface uses only letters. Their maximal length is 255 bytes (see grade of service = basic) from which 52 are used for administration. The longer messages are truncated.

3.3. Input from DCAM applications

The letter arrival is announced by a "LETTER ARRIVED" event queued on the ROUTING-bourse. NBROUTE tries to receive the letter ($RECLET) and analyses the return code provided by BCAM. It distinguishes three groups :

1. the return code is "OKAY" or "LETTER LENGTHER"; then, the input is correctly terminated.

2. the return code is "TRY LATER"; then, NBROUTE waits one second and retries to receive the letter. After five successive "TRY LATER" return code, it simulates a permanent error.

3. the return code takes one of the possible other values; then, NBROUTE assumes a permanent error

In permanent error case, NBROUTE assumes the letter lost but does not signal any defective state for the sending application.

Note that he sender continues its work asynchronously without waiting for the acknowlwdgement from NBROUTE.

3.4. Output to DCAM applications

Letters destination may be given by a routing code (communications) or an authorization name (communications and special commands). NBROUTE tries to send the letter ($SENDLET) to its destination(s). For each letter sent, it checks the return code provided by BCAM,it distinguishes three groups:

1. the return code is "OKAY" ; then, NBROUTE assumes a successful send.

2. the return code is "TRY LATER"; then NBROUTE waits one second (PASS) and retries. After four unsuccssful retrials, it simulates a permanent error

3. the return code belongs to the other possible values; then, NBROUTE assumes a permanent error.

In permanent error case, NBROUTE tries to send the communication to the main console (implicit simple replacement handling) if its destination was an authorization name and forgets the output otherwise.

4. Logging

4.1. Initializations

The first baby's cries of a new system are processed by a STARTUP module using the emergency messages interface as the normal messages handler does not yet exist. Thosse messages are logged, in a given format, in a logging buffer (length 12KB).

When the UCON-task is activated by E2STARTU, it creates a NON PREALLOCATED logging task (TSN = CLOG) and its input interface: the logging bourse.

The further operating communications are stored by the CLOG-task in the logging buffer according to an other format. As soon as the Data Management System (DVS) is available, CLOG

- opens a dynamically expanding file (access PAM) called SYS.CONSLOG.YY.MM.DD.###.SS
- signals the event to the consoles
- transfers there the buffer content (the communications reaching CLOG during this time stay queued on the logging bourse)
- frees the logging buffer memory space

4.2. Normal Logging

CLOG stores directly the further communications in the file. Between two accesses, the logging file stays opened, this is time sparing but not so secure.

4.3. Logging File Changes

During a session, there may be at most 98 logging file changes bound to the /CHANGE-CONSLOG command processing or file crashes (DVS) handling.

If less than 98 changes were already completed during the current session, the change is completed in three steps:

1. The old file is closed after time stamp addition. The associated message (E657)is sent t the consoles and the old file name is stored in a special file named "SYSCONSLOG.TRANSF.###".

2. A new logging file is opened and the associated message (E040) sent to the consoles. The two first logged messages correspond to the closing of the old and the opening of the new logging file. (E657 and E040)

3. The old file is converted to a SAM (sequential access) file to allow further access.

If 98 changes were already completed, a /CHANGE-CONSLOG command processing is rejected. At this moment, if a Data Management System (files) failure occurs, this event is announced to the CLOG-task which sends the appropriate message to the console, stops the logging and only releases the further SQEs.

4.4. Logging File Informations

The operator can issue

- /SHOW-CONSLOG command to know the current logging file name.
- /TURN command (one at a time) to read in the current logging file.

Those commands are processed by CLOG-task modules.

4.5. System Shutdown

The CLOG-task can be told about a shutdown in progress. Then, simulating its completion, it logs in the logging file the shutdown message (E557), closes this file as described in the first step of the file change and sends the associated message (E657) to the consoles.

This last message logging and the SAM conversion can be completed during the next STARTUP if the operating system version is the same.

4.6. Logging Task Crash

If, fo any reason, the CLOG-task terminates abnormally, its partners (ECTYP, UCON,OPR) are not informed and no restart is provided since CLOG is not a preallocated task. The further SQEs are released neither by the crashed loggingtask nor by its partners since logging is not completed. Therefore, the system turns slowly bu surely to a memory class 3 space shortage that finally leads to a system crash.

SECTION 2 :

NEW CONSOLE OPERATIONS CONCEPTS FOR LARGE CENTRALIZED COMPUTER SYSTEMS

In order to prepare the future, know about the past. This should be a proverb for information system developer. Let us apply it in the following according to this sequence :

1. Concepts Evaluation
2. Design Criticisms
3. Requirements Compilation
4. Introduction of new concepts

Chapter 5 : Concepts Evaluation

One can find in the following a private approach to describe the dissatisfaction causes in the current console operations subsystem. It states the CONCEPTUAL LEVEL lacks of the current solution trying to show their double significance :

1. At the user level

More and more, it is necessary to approach as finely as possible aggregate requirements from users. Existing users first since they know rather well the strong and weak points of the current products. They constitute all of the manufacturers current rental base and about two thirds of their total market for new sales. Possible future users in a second time since gaining new customers is something very hard on the computer market.

Consequently, it is interesting to find and modify appliances that are not optimal (from user-friendliness to crash problems) for some "virtual user".

2. At the system level

It is well known that earlier an error occurs in the software development process, more important are its consequences on the final product. Our interest relies here to study the consequences of conceptual lacks on the current design with regard to qualities often stated for large-scale software final products such as

- RELIABILITY : effectiveness, robustness, availability, ability to test or verify and to diagnose.
- MAINTENABILITY : ability to modify, extend or contract the product considering that the repping policy is not the most appropriate way to change software.
- SECURITY : protection of data and programs.
- EFFICIENCY : at a global level.

1. System Management

The development of the system management interface has followed in the past generations a random path marked out with serendities and technical constraints more than organizationnal requirements. Its resulting division in system administration and operating is characterized by a polarization of power and competence.

The administrator centralizes, indeed, many duties which scope stetches from technical questions like system control OUTSIDE the operating session (see chapter1, 3.4.2 page 1.8) to pure administrative ones like accounting. To complete them, the administrator should be a super (and expansive) multi-skilled individual. In practice, however, he is often more limited and his technical duties often overshadow the user guidance ones. In this way, the current management division contributes to the bad image of the computer center through the organization. The communication gap between computer experts and the line management follows even a growing trend with technical progress, creating sometimes important problems.

As opposed to this administrator felt as the nerve center of the computer system, the operators tasks are subqualified. Their hands-on standardized nature does not increases the motivation inside the computer room. Furthermore, this subqualification should be coupled in a more or less nearby future with a disqualification process reducing to an irreductible level the peripheral devices operators number if the trend toward devices manipulations simplification keeps going on.

In practice, if the tasks distribution is not always so contrasted, it is often paid by a lack of clarity and control possibilities.

2. Console Operations

The current console operations do not involve utility programs execution although they are necessary to the good working of the system. Operators are then allowed to work on the system as user and sometimes even as administrator from a local terminal. Consequently, an operator with some knowledge of software and/or hardware and "trapdoors" or weaknesses is invariably placed in a unique position of control. More simply, the temptation to play with the system (under the TSOS userid if allowed) during his idle times is sometimes too big for the operator. Clumsinesses can then lead to catastrophic situations.

The special NDCATOP interface, used to give the console operator the possibility to issue certain system administrator commands, does not solve this problem since the /EXEC command is not allowed.

3. Operating Sources

The current operating sources concepts stay based on an exclusively LOCAL operating philosophy. Since the authorities control was copleted at human level in the computer room, the system did not have to distinguish among operators and it could so assign rights to the devices.

In a future remote operating philosophy (at least from remote terminals), it seems not secure enough to still keep consoles as authorities distribution subjects since human visual control disappears. The local/remote transparency should stretche this requirement to all operating forms.

The console operations subsystem is characterized by operating source dependance inside and independance outside.

From one hand, indeed, it is spangled with hardware appliances which reappear here and there introducing exceptions like the teleservice buffer. Authorized applications follow a specific path (without replacement handling..).

From the other hand, all inputs, whatever source they come from, are bound to the same format so that authorized applications authors have to put TEXT ORIENTED statements in their (assembler) programs. This should not simplify the future development of automatic operators.

4. Competence Distribution

The competence distribution as previously described (see page 2.5) merges together ROUTING and AUTHORIZATION codes. This correlation is not optimal. Moreover, the "x" routing code is not always consistent.

4.1. Competence Area

The current competence area plays two parts. The first one is the AUTHORIZATION AREA : the maximal and theoretical competence area assigned by a key individual (authorizer). As the messages flow is sometimes too important, an operator can reduce it by tuning a WORKING COMPETENCE AREA. This two-fold nature of the area is illustrated by the possible actions on it.

4.1.1. Main Console : Authorization

The main console operator is a kind of authorizer for operators. He uses

- /CONSOLE SWITCH command to begin or terminate the work from a given console.
- /ASR (DELETE, ADD, PRIMARY) command to modify other operating sources competence area.

4.1.2. All Operating Sources : Flow Control

This tuning can involve

- the two sets : with the /ASR (DELETE, ADD or PRIMARY) command, an operating source can modify its own competence field. It can extend its area to a more important than the one allowed by the main console (?)
- the messages set : all questions, replies and single destination information messages must be received while other information messages can be filtered. The / ASR INF!NOINF command acts on the whole flow but it is also possible to fix a weight under which messages are not delivered.

A clear distinction between authorization and flow control, specific commands for authorization by the main console and a powerful and integreted tool for the flow control should be improvements in future solutions

4.2. The "*" Routing Code

The "*" routing code concept is inconsistent. Theoretically, it is assigned to an only to the main console. In practice, however, any opering source may be assigned it at the system generation or using the /ASR ADD command. Some scattered tests try to correct this problem but they are not transparent.

5. Messages

One can raise the following notes about messages reaching the operating sources :

1. Real destination uneasy to find
2. Correlation messages/initiating event sometimes uneasy
3. Messages hybridity
4. Standard messages freedom

5.1. Messages real Destination

A priori, the question or information message sender has few means to anticipate clearly the actual destination of his communication. One can see here the critical influence of the coding on the functional level :

1. It seems difficult to give the operator functional rules allowing him to understand the system behaviour.

2. Any coding change (even procedures inversion) should be catastrophic. (in the best case, functional specifications changes; in the worst, errors)

3. There are hidden implications. For instance, a console can receive messages even if it does not want them (see control flow) since the filtering is made before the replace console handling.

A posteriori, the authorized application sender receives no acknowledgement. The console sender receives an acknowledgement from the console operations subsystem (NBROUTE) but no one from the actual receiver(s)

5.2. Correlation messages/initiating Event

The same message corresponds sometimes to quite different situations (e.g. REJ7 USER NOT ALLOWED TO REPLY)

The concept of messages set associated to an event does not exist, this is mainly perceptible in

- the replace console handling : a state change triggers several actions, an isolated message is sent for each action (sometimes to different partners)

- the commands processing : as commands processors send command processing communications separately, those one can be mixed with the other communications.

This problem is augmented by the communication maximum size limitation. Pictures from the /STATUS command processing, for instance, must be divided in several messages. Sent separately, they are mixed with the normal communications flow.

5.3. Messages Hybridity

The messages reaching a given operating source can have different formats mainly depending on their source and they are described in different reference manuals. The following picture tries to give an overview about them.

recipient	format	id.	source	ref. manual
console	*** <TEXT>	W.T.	KTT	[BEDI75]
auth. app.	data orien.	R.C.	BCAM	[DCAM70]
all	REJ_i <TEXT>	REJ_i	UCON	[BEDI75]
	NB (TE,IN_j)*	NB and IN_j	SYSTEM or USERS (S)MSG(7)	[SYSM76] inserts ?
	<TEXT>	W.T.	SYSTEM or USERS TYPE(IO)	none

id. = identifier
W.T. = whole the message
R.C. = return code (see [FRIE80])
TE = part of message text
IN_j = insert
NB^j = standard message number

Such an hybridity is not so user-friendly for the human operator. Coupled with the inserts problem (their number, position and content is defined formally nowhere) it reduces the automatic operators possibilities.

5.4. Standard Messages Freedom

Nothing identifies standard messages sent by system modules from standard messages sent by users or free messages with exactly the same format as standard ones. Consequently, when the human operator receives a critical message such as "SYSTEM FAILURE. PLEASE COMPLETE SHUTDOWN", he has to check carefully its source to prevent stupid jokes. But what about an automatic operator ?

6. Commands

The lack of commands completion acnowledgement imposes sometimes the operator to have faith in the system (an automatic operator too ?).

No acknowledgement is provided for commands issued from authorized applications. The acnowledgement to console comes after partial checks in order to process normal and special commands in the same way as long as possible. (since special commands operands are not defined at the system generation). Since then, they are regarded as completed so that several commands issued from the same console can be in progress in parallel. This poses synchronization problems if the result of the second command depends on the first one completion. Furthermore, the mixing of the commands processing communications is not so user-friendly.

7. Special Replies

Replies from tasks are considered by now as exceptions. The provided interface, testifying it, can be improved.

In the future, if the actual trends toward parameterization (PRIOR), system self-control and -tuning and on-line diagnosis possibilities keep going on, requests to the responsible modules should be commonplace. Consequently, replies from tasks conceptualization deserves to be got to the root

8. Replace Console Handling

The replace console handling is triggered by consoles state changes at the two - logical and physical - levels. The actual association is perhaps not optimal.

At the physical level, not available states involve the situation of the device (defective or not attached) without anticipating the operator's willingness. In general, the device failure interrupts the work of the operator (this allows often its identification) who wishes to continue it from an other console. The replace console handling is adapted to complete this transfer.

At the logical level, however, the console switch off seems to be related to the termination of an operator's work. The transfer of competences is then motivated more by system reliability requirements than the operator's willingness to continue his work. This device independant break seems to belong to the same family as the disconnection for an authorized application. The replace console handling provides a tool to insure the system reliability but it seems

- EXPENSIVE : the actual replace console search is sometimes expensive and it is not necessary if the operator duties are assumed from other sources.

- UNFAVORABLE TO CONSISTENCY : it encourages the trend toward duties scattering while we think that consistency (control unicity or explicit concurrency handling) is an important feature in multiple consoles and remote configuration. As concurrency between operators seems to be difficult to control, we propose to reduce it. (1)

---

(1) This problem is only theoretical in the current situation since operating is local and the consoles actual number remains little (4).

Chapter 6 : Design Criticisms

By care for completeness, we state in the following criticisms involving only design decisions.

1. Bourses

Communications (specially commands) are delayed in several (perhaps not always necessary) bourses :

- ROUTING bourse
- COMMAND bourse
- KTT bourse
- LOGGING bourse
- Command Processor Tasks bourses

2. ECTYP interface

The use of the same (ECTYP) interface suspending all communication sending tasks in the same way reduces some system tasks (OPR) availability and increases the deadlock risks.

3. $CONSOLE Interface Problems

Are listed here after the most significant problems involving the $CONSOLE application.

3.1. The Special Commands Problem

Special commands processor tasks are DCAM applications. The command transfer in their direction is completed exactly in the same way as single destination communications sent to authorized applications. That means that

1. In PERMANENT ERROR case, the command is sent to the main console. The question now for the operator is how to react to this event.

2. A special command sent is assumed to be correctly received. If it is not so, nobody knows it.

3.2. The "SHORTAGE OF RESOURCES" Problem

Two (limited) BCAM buffers are dedicated to each connection (one by partner) as the partners may be in different host computers. They are used to store letters until their announcement receipt. In sending, the letter is kept in the sender associated buffer for possible retries. (see DATA LINK LAYER procedures like HDLC...). In receiving, the letters (possibly many if there are partner's failures (cycle..)) stay in the buffer until receipt.

If the buffer associated with the UCON task becomes full, all attempts to send letters are rejected with the return code "Shortage of Resources". The problem now is how to react in such situation.

If the full state is bound to a particularly important flow between the partners, it will be solved with the time.

Let us consider an other situation. When the UCON task receives a letter announcement, it makes maximum 5 attempts to access the letter after wich it considers as lost the letter that starves in the BCAM buffer. There are two possibilities :

1. The BCAM task releases the memory space after a given time-out and the "Shortage" problem is only temporar.

2. The letter stays starving in the buffer. As such situation can repeat itself, the associated buffer becomes slowly but surely full only with those garbage letters. At this moment, the "Shortage" state is permanent. No more communications may be sent the application which can be considered as dead and all the future messages are lost. The only way to come back to a normal situation is to issue a /SHUTDOWN command and a new startup.

3.3. Deadlock Problems

The IBCAM [FRIE80] macro's used to administrate the $CONSOLE interface are first processed by several modules running under the caller task TSN. Let us call them the IBCAM modules.

This ones "pend" the appropriate orders on the BCAM task bourses. The calling task is suspended according to the synchronous use of the bourses.

The UCON task is bound to the same rules; when it issues an IBCAM macro, it is pended and,since then, all ECTYP uses become dangerous. For instance, if BCAM asks a question at this moment, it is also pended waiting for UCON processing and the system reaches a deadlock state. This is also the case if BCAM shuts down when a UCON call for an IBCAM macro is in progress.

4. Logging Task Crash

If, for any reason, the CLOG-task terminates abnormally, its partners (UCON, OPR) are not informed and no restart is provided since CLOG is not a preallocated task. The further SQEs are neither logged nor released (since logging is not completed). Therefore, the system turns slowly but surely to a memory class 3 space shortage that finally leads to a system crash.

Chapter 7 : Requirements Compilation

People often state ease of use, reliability, serviceability and nondisruptive growth as main qualities for a piece of software. We make them ours for the console operations subsystem but primary influences such as anticipated user requirements and operating philosophy trends lead us to the adoption of some more specific objectives such as

- Clear management functions distribution allowing some flexibility but showing explicitly the powers trusts.

- A commands language providing consisten access to all control programs and utility functions necessary to the operating.

- High levels of device independance provided to operators including screens formats, communications formats (text or data oriented) and local/remote transparency.

- High levels of integrity and authorization facilities with minimal corresponding overhead.

- Clear matching between the console operations subsystem and the operated system (effectiveness).

In order to eliminate potential problems of design mismatch, it seems interesting (if not necessary) to impose NO COMPATIBILITY CONSTRAINTS with previous solutions. Although we understand the commercial impact of the compatibility argument, we persevere because

- A new console operations subsystem involves only few specialized individuals in the organizations and does not disqualify the expensive software investments (by opposition with the user level changes). Furthermore, the new solution should be more user-friendly so that the immediate negative impact of uncompatibility is no so important.

- The operating philosophy is changing now from a local hands-on interactive nature to a mixed nature local/remote human/automatic. If the console operations subsystem does not follow this trend, perhaps changing completely, it may well be either old-fashioned or unmaitenable in some years.

Consequently, the money lost immediately could be an investment which benefits will be great in some years.

- It is more motivating to develop properly and maintain a new piece of software than to maitain an old "tricky" one.

The extensive capabilities described for a new subsystem are made possible in the "no compatibility constraints paradigm" by the definition of novel concepts, their traduction in an advanced design and a new implementation of system components, software and perhaps hardware. The following chapter introduces some new concepts allowing to fulfill the first step of this process.

Chapter 8 : Introduction of new Concepts

1. The initial Concept : the Mail System

Schematically, the console operations subsystem can be seen as an electronic mail system. It connects, indeed, one to the others different kinds of partners :

- Operators
- Users
- The system in which we distinguish
  - normal system tasks
  - normal commands processor tasks
  - special commands processor tasks
  - tasks authorized to reply

The operator duties are completed inside a so-called operator session which can be initiated either directly using a special command or from another session requesting a utility program execution or a command file processing "in background". It is eventually interrupted either explicitly to allow the operator to act as user or implicitly by the execution request for a utility program if the operator want to monitor it interactively. It is terminated either because of an explicit command or the special duty termination (EXEC or RUN).

Each operator is defined a profile which describes hi his rights. His actual working area depends on his profile as well as on his willingness (flow control) and the properties of the console he is working from.

The session is made of conversations called transactions between an operator and one or more other partner(s). A transaction is a semantically coherent set of informations wich smallest piece is called communication :

- information message
- question
- reply
- command
- acnowledgement

Some conversations occur in special conditions and have to be handled in priority. Let us call them emergency transactions

2. The night-light console concept

This initial concept is not always compatible with the reality. Some harware requirements such as setup, cpu control and devices manipulations impose human presence inside the computer room. Furthermore, if today's systems are no more operator-bound, some situations require still fast reactions.

To reach these problems, we define a night-light console wich must be a local console. The associated operator session is equal to the system one, it can not be interrupted (from which its name). Its operator acts as

- garbage collector or watch dog : he has no personnal rights but collects all those left by operators out of sessions. In problem case, he may react or warn a more specialized operator.

- hardware monitor : specialized in hardware question he assumes the startup, some emergency situations (shutdown because of hardware failures), physical I/O retries requests, self-loaded programs monitoring, cpu tests...

This concept is not optimal with regard to the first kind of functions but, if the system self-control trend developes, the number of such cases can be reduced and a ringing watch-dog (beep machine) could be suffisant.

3. Operator

We distinguish three kinds of operators :

3.1. System Manager

His duties combine the system control duties of the administrator to the computer operator ones as th they were described in the current solution. Furthermore, he is the authorizer for possible specialized operators.

3.2. Specialized Operators

They are perpheral devices operator specialized in some given system fields. Their number and the number of consoles they work from depend on the computer system size.

3.3. "Night-light" Operator

They are specialized in harware questions

4. Profile

The profile defines for each operator

- a competence area : it defines the set of messages of interest for the operator and can remain based on the routing code concept as previously described

- a command area : it defines the set of authorized commands. This set is independent of the first one. Such a definition requires a fine and granular commands structure (a tree with nodes as marks for instance).

- a utility area : it defines the set of available utility programs.

5. Operator Session

There are two kinds of operator sessions :

1. Special purpose one : They are triggered from an other session to complete some specific tasks (utility program or commands file) and are not involved by the current communications flow.

2. General purpose session.

6. Logical Console

Inside the session, the operator is known by the system via a logical console. Without anticipating its physical features, it is defined by the following logical features :

1. Identifiers

- mnemonic name
- number

2. Commands Area

Area reserved to the commands transactions; commands in input and processing communications in output.

3. Messages Area

Divided in an outstanding questions area and a messages area.

4. Logging Area

Used to keep track of all communications received by this console in their chronological sequence.

All these areas are optional. If they are not defined, the corresponding duties are not allowed from the console (independently of the physical ability)

7. Transactions

A transaction is a conversation between partners among which there is an operator. We distinguish different kinds of transactions :

7.1. Commands Transaction

The commands transaction is initiated by a command sent by the operator to a command processor task. It is abnormally terminated by a negative acknowledgement from the system (syntax, authorization checks...) or a negative receipt acnowledgement from the processor (time out, other critical command in progress..).

If the command reaches the processor task, the second part of the transaction begins. It is materialized by an information message chain called command processing communications chain. This can contain a question, then a question-reply transaction is contained in the other one. The command processing communication chain is finally terminated by a command completion acknowledgement.

7.2. Question-reply Transaction

A question is received in a first time by the console operations subsystem (with acknowledgement) and in a second one by the operator (with ack.).

The operator can answer it sending back a reply chain or reverberating it to the authorized task. Thi This task will eventually send the reply chain. The end of the reply chain is also the end of the transac tion.

7.3. Messages transaction

A messages transaction is made of a messages chain in one direction and the corresponding receipt acknowledgement in the other.

7.4. Emergency Transaction

An emegency transaction is a messages transaction or a question-reply one processed with bigger priorities.

8. Messages

It is made of two part : the head and the body

8.1. Head

- number
- source identifier
- time
- destination identifier : routing code or mnemonic name or operator identifier
- weight
- secret ?
- message nature : ack. standard free

8.2. Body

- number of parameters
- list of parameters
- text

9. Messages Filtering

It is based on the routing codes and/or the messages weight.

Conclusion

Often neglected in the last generations, the system operating stands today at a crossroads. From one hand, indeed it is still highly influenced by the batch operating paradigm as its hands-on interactive nature can testify. Therefore, its delay with regard to user-level appliances seems to be very important. From the other hand, operating should evolve quickly in the nearby future led by user requirements (see the introduction of the computer in unsophysticated circles). Some authorities state that those rapid progresses will be coupled with the operating splitting in different forms : CENTRAL OPERATING FROM COMMONPLACE TERMINALS, AUTOMATIC AND REMOTE OPERATING...

The hypothesis of this work was that the current console operations subsystem does not provided guarentées to serve as reliable root for the future building of the operating "tree" because of conceptual lacks. The first section supported this hypothesis by a case study.

Consequently, the thesis claims for the use of conceptual requirements as directional guideline for the console operations subsystem design an future maitenance. The second section identified the old-fashioned or weak concepts and proposed to replace them by new ones without taking compatibility constraints into account. This choice for freedom is assumed as an investment (perhaps expensive) which should pay back benefits in the long term.

An important step is completed insofar as the console operations subsystem problem is circumscribed and some ideas of solution provided but it is just a little step since the new concepts must be refined and faced with the real system life in order to result in a good design and, finally, in an operationnal implementation. The console operations subsystem is on the road but its travel is still long.

BIBLIOGRAPHY

ADSB84 Manual :
"ADS (Transdata PDN). Benutzerhandbuch V1.1A"
Siemens, Munich, 1984

ARIT80 I. Arita :
"Intelligent Console. A universal User interface of a Computer System"
Faculty of Engineering, Kyushu University, Fukuoka 1980

BEDI75 Manual :
"BS2000 Bedienunganleitung V7.6A"
Siemens, Munich, 1984

BENE75 Hr. Benedickt :
"Ersatzkonsolbehandlung"
BS2000 V7.5. 22-52-4212
Siemens, Munich

BIRD75 R.A. Bird, L.A. Hofmann :
"System Management"
IBM System Journal, Vol 19, No 1, 1980, 140-159

CLEM80 E. Clemens, K. Gallasch :
"Filterung von Konsolmeldungen"
BS2000 V8.0. 043-52-0920
Siemens, Munich

CLEM90 E. Clemens :
"Logische Konsole"
BS2000 V9.0. 043-41-2015
Siemens, Munich

DCAM70a Manual :
"Transdata BS2000 DCAM Programmschnittstellen V7.0"
Siemens, Munich, 1982

DCAM70b Manual :
"Transdata BS2000 DCAM Makroaufrufe V7.0"
Siemens, Munich, 1982

DEMO75 Ph. Demoulin :
"L'Informatique : une Entreprise à gérer"
Institut d'Informatique, Namur, 1975

DEMU84 M. Demus :
"Vorschlag für die Struktur eines einheitlichen Administration-Konceptes für BS2000"
BS2000. 43-36-748
Siemens, Munich, 1984

DEVE80 Manual :
"Developers Handbook"
BS2000 V8.0. 500-52-99
Siemens, Munich

EINF75 Manual :
"BS2000 Einführung in die Systembedienung. Benutzerhandbuch V7.5A"
Siemens, Munich, 1984

FINE83 L. H. Fine :
"Computer Security. A Handbook for Management"
Heinemann, London, 1983

FRIE80 Dr. Friedl :
"DCM IBCAM. BCAM-user interface"
BS2000 V8.0. 554-52-61
Siemens, Munich, 1983

GALL85 K. Gallasch, A. Seiler :
"CFCS-3 Konsoleinstellung : Kommandos"
BS2000 V8.5. 043-52-1020
Siemens, Munich

HENR78 G. G. Henry :
"Introduction to IBM System 38 Architecture"
IBM System/38. Technical Developments.
IBM Products Design and Development.
General Systems Division.

KASE80 Hr. Kasek :
"Nucleus Device Management (NDM)"
BS2000 V8.0. 022-52-21
Siemens, Munich

KATZ80 H. Katzan :
"Operating Systems : a pragmatic Approach"
Van Nostrand Reinhold company, New York, 1980

KIND81 R. Kinderlehrer :
"Handbook for Data Center Management"
Q.E.D. Information Service, Inc.
Welleslay, Massachussetts, 1981

LEAC80 J. R. Leach, R. D. Campenni :
"A Sidestream approach using a small processor as a Tool for Managing Communications Systems "
IBM System Journal, Vol 19, No 1, 1980, 120-139

LIU 84 T.S. Liu :
"Maintenance processors for Mainframe Computers"
IEEE Spectrum, feb. 1984, 36-42

MORG79 L. Morgan :
"Managing on-line Data Communications Systems"
NCC Publications, Manchester, 1979

OMNI75 Manual :
"Transdata BS2000 Omnis Benutzerhandbuch V4.1"
Siemens, Munich, 1983

RIMK90 J. Rimkus :
"Verfuegbarkeit des Operatortasks"
BS2000 V9.0. 043-41-2016
Siemens, Munich, 1984

RAMA82 J. Ramaekers :
" Systemes d'Exploitation"
Notes de cours.
Institut d'Informatique, Namur, 1982

SEIL85 A. Seiler :
"CFCS-3 Konsolbedienung in V8.5"
BS2000 V8.5. 043-42-1020
Siemens, Munich

SEVP82 Manual :
"Service Processor, TELESERVICE
Siemens Systems 7500
Siemens Systems 7700
Beschreibung"
Siemens, Munich, 1982

SYGE75 Manual :
"BS2000 Systemgenerierung Beschreibung V7.5A"
Siemens, Munich, 1984

SYSM76 Manual :
"BS2000 V7.6A Systemmeldungen"
Siemens, Munich, 1984

TRAN96 Manual :
"Siemens-System TRANSDATA 9600.
Allgemeine Beschreibung"
Siemens, Munich, 1983

UBER75 Manual :
"BS2000 Systemüberwachung V7.5"
Siemens, Munich, 1984

VERD83 E. Verdier :
"La Bureautique"
La Découverte, Paris, 1983

Lectures notes from

BS20P BS2000 Systembedienung
03.12.84 - 14.12.84

BS2SV BS2000 Systemverwaltung
26.11.84 - 30.11.84

TN-BS2 Teilnehmerbetrieb im BS2000
29.10.84 - 31.10.84

Models extracted from

BODA83 F. Bodart, Y. Pigneur :
"Conception assistée des Applications informatiques.
1. Etude d'Opportunité et analyse conceptuelle."
Masson, Paris, 1983

APPENDICES

Appendix 1 : Model of the dynamics of processing

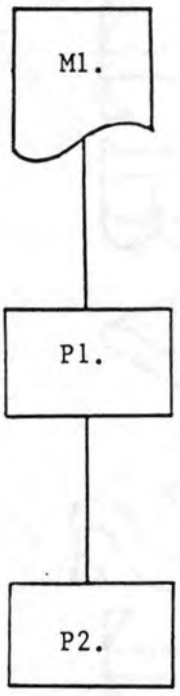
Le modèle de la dynamique repose sur deux concepets de base:
le processus et l'évènement.

Un processus est l'exécution d'une procédure de traitement de l'information dont la progression peut être représentée, à des points dans le temps, par son état (état déclenché, actif ou terminé).

Un évènement correspond à un changement d'état du système d'information localisé dans le temps et dans l'espace.
Un évènement est dit externe s'il correspond à l'apparition d'un message qui déclenche un processus du S.I.
Un évènement est dit interne s'il correspond à un changement d'état interne au S.I.

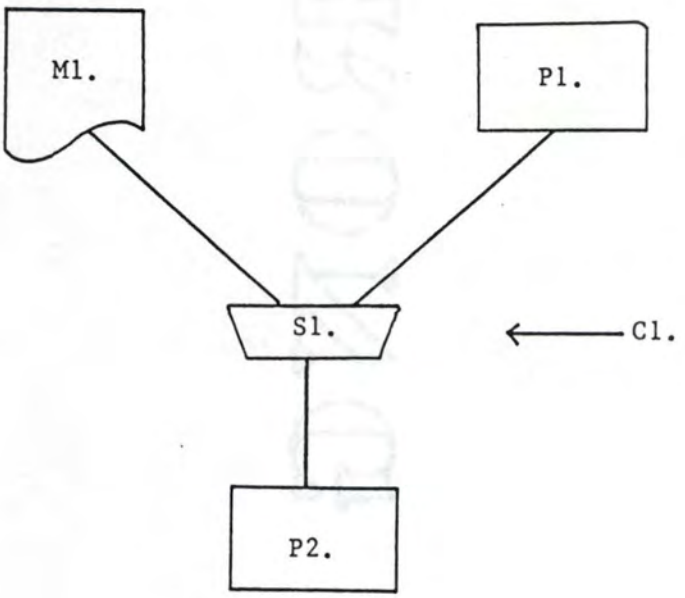
Nous allons décrire à présent différentes structures dynamiques que nous rencontrerons dans le modèle dynamique de notre système d'information.

- <u>structure séquentielle classique</u>



L'apparition du message M1. (évènement externe) déclenche le processus P1. La terminaison du processus P1 (évènement interne) déclenche le processus P2.

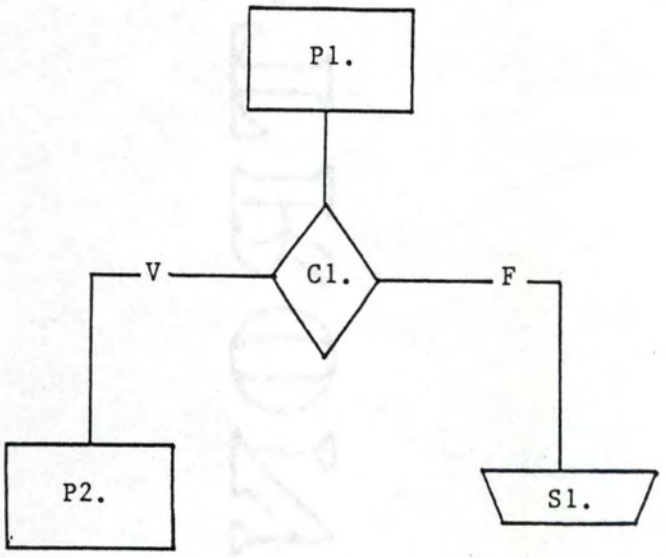
- structure de synchronisation



L'apparition du message M1. (évènement externe) contribue au point de synchronisation S1. (mécanisme de coordination d'évènements). La terminaison du processus P1 (évènement interne) contribue également au point de synchronisation S1.

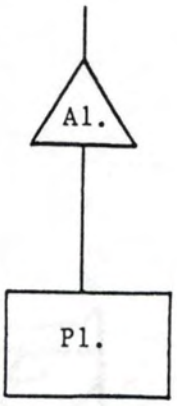
Au point de synchronisation S1. est associée une condition C1. Le prédicat de cette condition est une combinaison logique des évènements qui contribuent au point de synchronisation S1. Lorsque le prédicat est vérifié, la condition est réalisée et provoque la réalisation du point de synchronisation S1. La réalisation du point de synchronisation S1. (évènement interne) déclenche le processus P2.

- structure d'éclatement



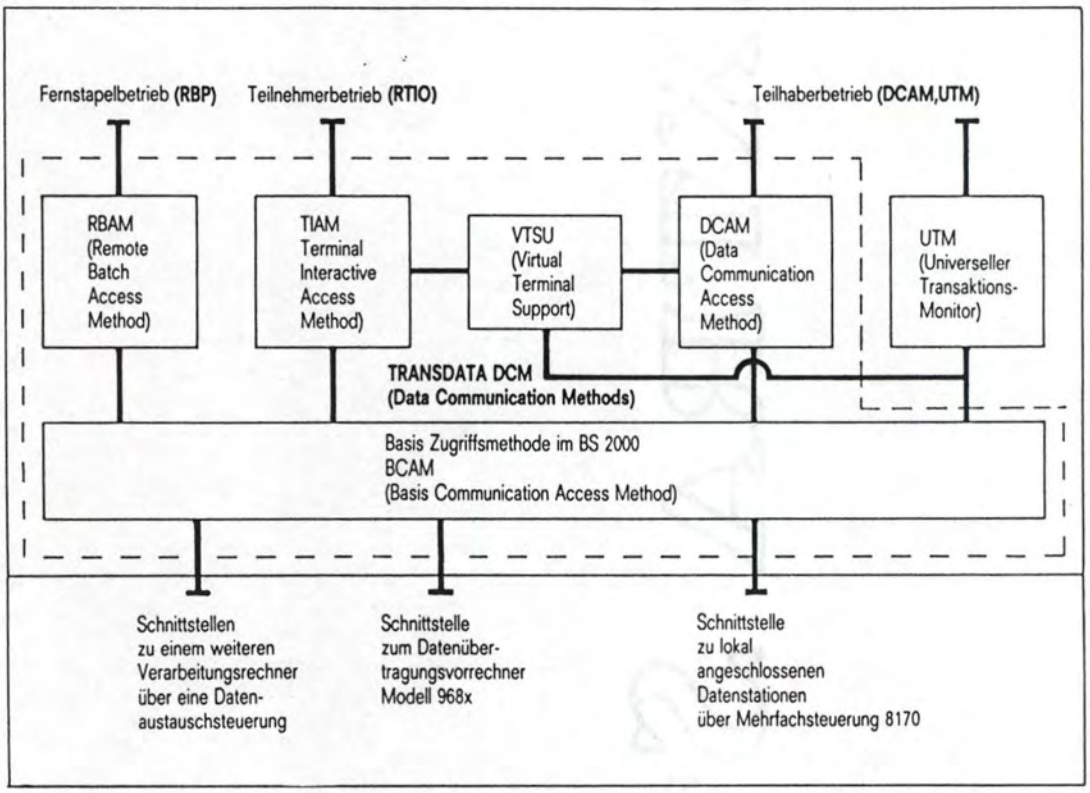
La terminaison du processus P1. (évènement interne) déclenche le processus P2 si la condition C1. est vraie ou contribue au point de synchronisation S1 si la condition C1. est fausse.

- structure itérative



Le processus P1. est déclenché autant de fois que la valeur de l'attribut A1. L'attribut A1. représente dans notre système d'information un volume de documents.

Appendix 2 : TRANSDATA



TRANSDATA : general architecture

Querverweise sind durch Kursivdruck gekennzeichnet

A

Administration: *[TRANSDATA-] Administration*

Administrator: *[TRANSDATA-] Administrator*

Administrationsplatz
(administration console) Einrichtung, den den Menschen als *Administrator* mit dem *Administrationszentrum* verbindet.

Administrationssteuerung
(administration manager) Instanz im *Verarbeitungs-* und *Kommunikationsrechner,* die die Aufträge des Administrationszentrums ausführt und diesem Betriebsereignisse meldet.

Administrationszentrum
(administration center) Partner des *Administrators.*
Es steuert den Betriebsablauf durch Aufträge an die *Administrationssteuerungen.*

Aufforderung zum Aufbau einer Verbindung
(connection request) Forderung eines *Kommunikationspartners* an das *Datenkommunikationssystem,* die *logische Verbindung* zu einem anderen *Kommunikationspartner* aufzubauen.

B

Befehl
(instruction) *(DIN 44300)*

Benutzerservice
(user service) Dienstleistungen für einen der beiden *Kommunikationspartner,* die *Nachrichten* übermitteln. Der Benutzerservice setzt sich aus *Stations-* und *Portservice* zusammen, die auf den jeweiligen *Kommunikationspartner* bezogen sind.

D

[DCAM-] Anwendung
([DCAM] application) *Kommunikationsanwendung,* die von mindestens einem *DCAM-Anwendungsprogramm* gesteuert wird. Sie wird in einem oder mehreren *DCAM-Prozessen* erzeugt.

[DCAM-] Anwendungsprogramm
([DCAM] application program) Benutzerprogramm, das die Dienste der Zugriffsmethode DCAM benutzt. Es steuert die *DCAM-Anwendung.*

[DCAM-] Datenübermittlungsfunktion
([DCAM] data transmision function) Durch *Befehle* und Meldungen ausgedrückte DCAM-Funktionen, die im Zusammenhang mit Senden und Empfangen von *Nachrichten* und Quittungen stehen.

[DCAM-] Ereignis
([DCAM] event) Eines aus einer Reihe von DCAM-spezifischen Ereignissen, die zur Koordinierung bestimmter Vorgänge im *Datenkommunikationssystem* verwendet werden. Es trifft vom Programmablauf zeitlich entkoppelt ein (= asynchron eintretendes Ereignis).

[DCAM-] Existenzfunktion
([DCAM] existence function) Durch *Befehle* und Meldungen ausgedrückte DCAM-Funktionen, die im Zusammenhang mit der Erzeugung und Auflösung von *DCAM-Anwendungen* stehen.

[DCAM-] Namen-Zuweisungsfunktion
([DCAM] name assignment function) Durch *Befehle* bzw. Kommandos ausgedrückte DCAM-Funktionen, die es dem Benutzer erlauben, die Benutzerprogramme unabhängig von aktuellen Parameterwerten wie DCAM-Anwendungsname, Partnername usw. zu erstellen.

[DCAM-] Prozeß
([DCAM] task)

Ein Prozeß, der durch einen expliziten DCAM-Aufruf eine *DCAM-Anwendung* eröffnet hat *(Primärprozeß)* bzw. sich einer bestehenden *DCAM-Anwendung* bekannt gemacht hat *(Sekundärprozeß)*.

[DCAM-] Statusfunktion
([DCAM] status function)

Durch einen *Befehl* ausgedrückte DCAM-Funktionen, durch die eine *DCAM-Anwendung* Information über sich und über *Kommunikationspartner* erfragen kann.

[DCAM-] Steuerblockfunktion
([DCAM] control block function)

Befehle der DCAM-Schnittstelle, die zur Erzeugung sowie der Manipulation von Steuerblöcken dienen. Alle DCAM-Makroaufrufe beziehen sich auf diese Steuerblöcke.

[DCAM-] Verbindungsfunktion
([DCAM] connection function)

Durch *Befehle* und Meldungen ausgedrückte DCAM-Funktionen, die im Zusammenhang mit Aufbau und Abbau von *logischen Verbindungen* stehen.

[DCAM-] Verteilcode
([DCAM] distribution code)

Innerhalb einer Eingabenachricht eine definierte Zeichenfolge mit der für die Verteilung der Nachricht innerhalb einer *DCAM-Anwendung* besorgt wird; die Codeposition in der *Nachricht* ist beliebig, ihre Länge begrenzt.

Dialognachricht
(dialog message)

Nachricht, die eine Antwort erfordert oder eine Antwort ist.
Die erste *Nachricht* ist bei UTM die Anfrage eines *TRANSDATA-Kommunikationspartners*. Damit beginnt er einen *Vorgang* oder setzt ihn fort.

Dialogschritt
(dialog step)

Teil eines *Vorgangs*. Er beginnt mit einer *Dialognachricht* an einen *TRANSDATA-Kommunikationspartner* und endet mit dessen Antwort.

Daten

(DIN 44300)

Datenfernverarbeitungssystem
= DFV-System
(teleprocessing system)

Gesamtheit eines Systems mit Datenverarbeitung und Datenkommunikation.

Datenflußsteuerung
(data flow control)

Kapazitätssteuerung auf dem Weg der *Nachricht* durch das *Datenübertragungsnetz*.

Datenkommunikationssystem
(data communication system)

Komplexe Einrichtung aus Hardware- und Softwareprodukten, die es zwei oder mehreren *Kommunikationspartnern* ermöglicht, unter Beachtung bestimmter Regeln *Daten* auszutauschen.

Datenquelle
(data source)

(DIN 44302)

Datensenke
(data sink)

(DIN 44302)

Datenstation
(terminal)

(DIN 44302)

Datenstationsbenutzer
(terminal user)

Mensch, der eine *Datenstation* benutzt, um mit einem *TRANSDATA-Kommunikationspartner Daten* auszutauschen.

Datenstationsrechner
(terminal computer)

Kommunikationsrechner, an den *Datenstationen* angeschlossen sind. In ihm laufen *Kommunikationsanwendungsprogramme* zur Datenstationssteuerung und Datenverarbeitung ab.

Datenübermittlung
(data communication)

(DIN 44302)

Datenübertragungsnetz
(data transmission network)

Summe der Hardware- und Softwareeinrichtungen, die die physikalische Übertragung der Daten von der *Datenquelle* zur *Datensenke* ermöglicht.

Datenübertragungsvorrechner
(local communication computer; front end processor)

Kommunikationsrechner, der direkt am E/A-Kanal des *Verarbeitungsrechners* angeschlossen ist.

Datenverarbeitungssystem
(data processing system)

(DIN 44300)

Digitales Rechensystem
(digital data processing system)

(DIN 44300)

E

Expreßnachricht
(express message)

Nachricht begrenzter Länge an eine *DCAM-Anwendung* oder an eine *Datenstation,* die mit höherer Priorität als Normalnachrichten übermittelt und zugestellt wird.

F

Format-Datenstation
(format terminal)

Typ einer *logischen Datenstation.* Die Datenstruktur wird durch Felder mit unterschiedlichen Eigenschaften gebildet.

Freilaufende Nachricht
(unsolicited message)

Nachricht, die keine Antwort erfordert und keine Antwort ist.
Sendet sie ein *Datenstationsbenutzer* an ein Anwendungsteilprogramm, erhält er eine Standardquittung von UTM. Sendet sie ein anderer *TRANSDATA-Kommunikationspartner,* entfällt die Standardquittung von UTM.

K

Knotenservice
(node service)

Dienstleistungen für die Behandlung von *Nachrichten* von und zu *Prozessorknoten.*

[Kommunikations-] Anwendung
([communication] application)

Instanz zur Verarbeitung von *Daten,* die zwischen *Kommunikationspartnern* ausgetauscht werden.

[Kommunikations-] Anwendungsprogramm
([communication] application program)

Verarbeitungsvorschrift zur Steuerung der *Kommunikationsanwendung.* Sie benutzt die Schnittstellen eines *Kommunikations-Zugriffssystems.*

Kommunikationspartner:

[TRANSDATA-] Kommunikationspartner

[Kommuniktions-] Protokoll
([communication] protocol)

Beschreibung der Übergabebedingungen und Übergabeformate zwischen gleichartigen funktionalen Schichten im *Datenkommunikationssystem (Benutzerservice, Transportservice, Netzservice).*

Kommunikationsrechner
(communication computer)

Rechner zum Aufbau von *Datenfernverarbeitungssystemen.*

Kommunikationszugriffssystem
(communication access system)

Menge der Software-Komponenten eines Betriebssystems, die den Anwendungen Schnittstellen zur Kommunikation bieten.

L

Logische Datenstation
(virtual terminal)

Modellvorstellung einer *Datenstation,* deren Funktionen auf die physikalischen Eigenschaften unterschiedlicher *Datenstationen* abgebildet werden.

Logische Verbindung
(virtual connection)

Zuordnung zweier *Kommunikationspartner,* die es ihnen ermöglicht, *Daten* miteinander auszutauschen.

M

mehrfach benutzbare DCAM-Anwendung
(sharable DCAM application)

DCAM-Anwendung, die von mehreren *DCAM-Prozessen* gleichzeitig benutzt werden kann.

N

Nachricht
(message)

(DIN 44300)

Netzknotenrechner
(remote communication computer; remote front end processor)

Kommunikationsrechner, der nicht direkt an einen *Verarbeitungsrechner* angekoppelt ist und dessen Aufgabe auf die Datenkommunikation beschränkt ist.

Netzservice
(link service)

Dienstleistungen, die sich aus *Knotenservice* und aus dem *Portservice* zusammensetzen, der auf andere *Prozessorknoten* bezogen ist.

P

[PDN-] Anwendung
([PDN] application)

Kommunikationsanwendung im *Kommunikationsrechner,* die die Schnittstellen des *Kommunikationszugriffssystems* im TRANSDATA PDN benutzt.

Port
(port)

Einrichtung für den Datentransfer zwischen einem *Prozessorknoten* und seiner Umgebung.

Portservice
(port service)

Dienstleistungen zum Transfer von Daten zwischen einem *Prozessorknoten* und seiner Umgebung *(Kommunikationsanwendungen,* Nahperipherie-Geräte, *Datenstationen,* andere *Prozessorknoten).*

Primärprozeß
(primary task)

Prozeß, der eine *DCAM-Anwendung* eröffnet und deren Charakteristika bestimmt.

Programm
(program)

(DIN 44300)

Prozessorknoten
(processor node)

Netzweit andressierbare Instanz im *Verarbeitungs-* oder *Kommunikationsrechner,* in der die Leistungen des *Transportservices* erbracht werden.

R

Rechner
(computer)

(DIN 44300)

Region
(region)

Teilgebiet des *Datenkommunikationssystems.* Sie enthält einen oder mehrere *Prozessorknoten,* die zum Zwecke der Adressierung aus der Sicht des *Transportservices* zusammengefaßt sind.

S

Sekundärprozeß
(secondary task)

Prozeß, der sich an eine geöffnete *DCAM-Anwendung* anschließt und deren Betriebsmittel mitbenutzt.

Station
(station)

Aus der Sicht des *Transportservices* netzweit adressierbare Endstelle des *Datenkommunikationssystems*.

Stationsservice
(station service)

Dienstleistungen für die Vereinfachung der *Datenübermittlung* durch Behandlung der *Nachrichten* von und zu *Kommunikationspartnern*.

T

Transaktion:

Vorgang

[Transaktions] Anwendung
([transaction] application)

Kommunikationspartner, an den *Transaktionsaufträge* gerichtet werden und der die gewünschten Vorgänge abwickelt.

[Transaktions] Anwendungsprogramm
([transaction] application program)

Verarbeitungsvorschrift, um die Anwendung zu steuern.
Es benutzt die Programmschnittstelle des *Transaktionssystems*.

[Transaktions] Auftrag
([transaction] job)

Anweisung an die Anwendung, einen *Vorgang* durchzuführen.

Transaktionscode=TAC
(transaction code)

Information, um einen *Vorgang* zu steuern.
Bei UTM dient der TAC dazu, die Anwendungsteilprogramme anzusteuern.

[Transaktions] Sitzung
([transaction] session)

Zeitraum, in dem ein *Kommunikationspartner* Aufträge erteilen kann. Er ist begrenzt durch Zuteilung und Rückgabe bestimmter Befugnisse.

Transaktionssystem
(transaction system)

Menge der Softwarekomponenten, die die Schnittstellen der Anwendung zur Umwelt bilden. Es steuert und überwacht *Vorgänge*. Es bedient sich der Leistungen des Betriebssystems, insbesondere des *Kommunikationszugriffssystems* und Datenbanksystems.

[TRANSDATA-] Administration
([TRANSDATA] administration)

Inbetriebnahme, Steuerung und Verwaltung der TRANSDATA-Komponenten eines *Datenfernverarbeitungssystems*.

[TRANSDATA-] Administrator
([TRANSDATA] administrator)

Mensch oder *Programm*, dem die *Administration* obliegt.

[TRANSDATA-] Kommunikationspartner
([TRANSDATA] communication partner)

Instanzen, die *logische Verbindungen* unterhalten und *Daten* miteinander austauschen.

Transportquittung
(transport acknowledgement)

Meldung über Abschluß oder Abbruch einer *Datenübermittlung*.

Transportservice
(transport service)

Dienstleistungen für den Datenaustausch zwischen *Kommunikationspartnern*. Er veranlaßt und kontrolliert den Transport der Nachrichten durch das *Datenübertragungsnetz* und verwaltet *logische Verbindungen*.

U

[UTM-] Anschlußprogramm
([UTM] linkage program)

UTM-Anwendungsteilprogramm, das vom System bereitgestellt wird. Der Benutzer definiert lediglich seinen Umfang. Es schließt das Anwendungsprogramm an den UTM an. Jedes Anwendungsprogramm enthält ein solches Programm.

[UTM] Anwendung
(UTM application)

Kommunikationspartner, an den *Datenstationsbenutzer, UTM-Anwendungen* und andere *TRANSDATA-Kommunikationspartner Transaktionsaufträge* richten können. Die Anwendung wickelt die gewünschten *Vorgänge* ab. Sie wird generiert und durch das Anwendungsprogramm gesteuert.

[UTM-] Anwendungsprogramm
([UTM] application program)

Verarbeitungsvorschrift, um die *UTM-Anwendung* zu steuern. Es benutzt die UTM-Programmschnittstelle und besteht aus dem *UTM-Anschlußprogramm* und den Teilprogrammen des Benutzers.

[UTM-] Anwendungs Teilprogramm
([UTM] application

Abgeschlossener Teil eines *UTM-Anwenderteilprogramms,* der ein Teilprogramm einer Anwendung bearbeitet. Es wird durch den *Transactionscode* adressiert. UTM startet das Teilprogramm, wenn dafür eine *Nachricht* vorliegt. Ein Teilprogramm führt höchstens einen *Dialogschritt* aus.

UTM-Datenstation
(UTM terminal)

Datenstation, die durch ihren Namen existiert. Sie entsteht bei der Generierung der *UTM-Anwendung* und entkoppelt sie vom *Datenübertragungsnetz:* Eine UTM-Datenstation kann für mehrere physikalische Datenstationen stehen.
Mehrere UTM-Datenstationen können für eine physikalische Datenstation stehen. Eine UTM-Datenstation kann für einen beliebigen *TRANSDATA-Kommunikationspartner* stehen. Die Zuordnung kann während des Betriebs per Administrationskommando geändert werden.

[UTM-] Dialogschritt
([UTM] dialog step)

Teil eines *Vorgangs.* Er beginnt mit einer *Dialognachricht,* die vom *Datenstationsbenutzer* oder einem anderen *TRANSDATA-Kommunikationspartner* einer Anwendung geschickt wird. Er endet mit der Antwort der Anwendung.

[UTM-] Transaktion:

UTM-Vorgang

[UTM-] Transaktions Auftrag
([UTM] transaction job)

Anweisung an eine *UTM-Anwendung,* einen *Vorgang* durchzuführen. Er enthält einen *Transaktionscode* und ggf. zu verarbeitende *Daten.* Er wird von *Datenstationsbenutzern,* von *UTM-Teilprogrammen* derselben oder einer anderen Anwendung oder von anderen *TRANSDATA-Kommunikationspartnern* erteilt.

[UTM-] Transaktions Sitzung
([UTM] transaction session)

Zeitraum, in dem ein *Datenstationsbenutzer,* eine andere *UTM-Anwendung* oder ein *TRANSDATA-Kommunikationspartner* einer *UTM-Anwendung* bestimmte Aufträge erteilen kann. Er beginnt, wenn die Befugnis dazu erteilt wird. Er endet mit Rückgabe dieser Befugnis.

[UTM-] Vorgang=[UTM-] Transaktion
([UTM] transaction

Abarbeitung eines in sich geschlossenen Auftrags durch die Anwendung, die dazu eines oder mehrere Teilprogramme benutzen kann. Er kann aus einem oder mehreren *Dialogschritten* bestehen. Die Betriebsmittel, wie Speicher usw., sind ihm zugeordnet.

Appendix 3 : The Bourse Mechanism

Modern operating systems are mostly organized as parallel processes. Consequently, concurrency is a dominating aspect of the structure and understanding of an operating system.

In the BS2000 system, a PROCESS is the smallest subject able to execute instructions. TASKS are the subjects for the distribution of system resources (memory, CPU..). The schedule of each task is defined by an ordered chain of processes with the top one as being currently executed. The length of the chain (at least one process) can be increased or decreased by attaching or detaching processes at or from the top.

When a problem is solved by more than one process, a well defined relation (and data flow) between those processes has to be established. The bourse mechanism provides a tool for mutual exclusion, serialization and synchronization of processes. The objects controlled by the bourse are REQUESTS for specific processes.

1. MUTUAL EXCLUSION

The mutual exclusion of requests for processes is realized by a CHAMBER of the bourse which may be occupied only by one request at a time.

"Everydaylife" example of mutual exclusion : only one car at a time is admitted to the "chamber" of the car wash station.

2. SERIALIZATION

The serialization of requests for admission to a chamber is processed by the enqueuing call. It works with specified priorities of the requests (zero default) and FIFO or LIFO enqueuing discipline among same priorities.

"Everydaylife" example of serialization : cars in a queue before the car wash station.

3. SYNCHRONIZATION

A bourse with more than one chamber additionally provides a tool for synchronization between requests for different chambers of the same bourse. Admittance to the chambers is granted only if there are requests "ready to enter before both of the chambers. Expulsion is synchonized similarly.

"Everydaylife" example of synchronization : a taxi stand with one queue for the cabs and another one for the customers. The stand is "busy" only if and when the both queues are not empty. The business is handled between the queues headers (cab driver and customer). After they come to an agreement, they dequeue and let their successors, if any, handle their business.

4. DATA EXCHANGE BY A TWO-CHAMBER BOURSE

Requests synchronized by a two-chamber bourse may additionally have a synchronized data flow from the announcer of the request to the processing requested for the other chamber.

TABLE OF CONTENTS

GENERAL INTRODUCTION

SECTION 1 A CASE STUDY : CONSOLE OPERATIONS IN BS2000

CHAPTER 1 WHICH COMPUTER SYSTEM ?

1. Hardware

2. The System Software : BS2000

3. People : System Management in BS2000

3.1. Data Preparation and Hardware Maitenance

3.2. Administration and Operating

3.3. Operating Session
3.3.1 System Generation
3.3.2 Operating Session
3.3.3 The Repping : pseudo-Generation

3.4. Administration
3.4.1 Users Administration
3.4.2 System Administration and Control
3.4.3 Security
3.4.4 Computer Center Master

3.5. Operating
3.5.1 Computer Operations
3.5.2 Manual Operations

CHAPTER 2 CONSOLE OPERATIONS CONCEPTS IN BS2000

1. Operating Partners

1.1. User

1.2. Operating Sources
1.2.1 Console
1.2.2 Authorized Applications

1.3. The System

2. Operating Sources Properties

2.1. Consoles
2.1.1 Hardware oriented Functions
2.1.2 Multiple Consoles Congiguration Handling

2.2. Authorized Applications

CHAPTER 4 CONSOLE OPERATIONS SUBSYSTEM : ARCHITECTURE

1. Normal Command Processing

2. Physical Console Handling

   2.1. Consoles input and output
        2.1.1 Input fron Console
        2.1.2 Output to Console

   2.2. Attach and Detach

3. DCAM Applications Handling

   3.1. General Administration
        3.1.1. SCONSOLE Application Handling
               3.1.1.1 BCAM ready Handling
               3.1.1.2 BCAM shutdown Handling
        3.1.2. Connections to SCONSOLE
               3.1.2.1 Connection Request
               3.1.2.2 Connection Break

   3.2. Letters Handling

   3.3. Input from DCAM Applications

   3.4. Output to DCAM Applications

4. Logging

   4.1. Initializations

   4.2. Normal Logging

   4.3. Logging File Change

   4.4. Logging File Informations

   4.5. System Shutdown

   4.6. Logging Task Crash

SECTION 2 NEW CONSOLE OPERATIONS CONCEPTS FOR LARGE CENTRALIZED COMPUTER SYSTEM

CHAPTER 5 CONCEPTS EVALUATION

1. System Management
2. Console Operations
3. Operating Sources
4. Competence Distribution
   - 4.1. Competence Area
     - 4.1.1 Main Console : authorization
     - 4.1.2 All Operating sources : Flow Control
   - 4.2. The "*" Routing Code
5. Messages
   - 5.1. Messages real Destination
   - 5.2. Correlation Messages/initiating Event
   - 5.3. Messages hybridity
   - 5.4. Standard Messages Freedom
6. Commands
7. Special Replies
8. Replace Console Handling

CHAPTER 6 DESIGN CRITICISMS

1. Bourses
2. ECTYP Interface
3. SCONSOLE Interface Problems
   - 3.1. The special Commands Problem
   - 3.2. The"SHORTAGE OF RESOURCES" Problem
   - 3.3. Deadlock Problems
4. Lôgging Task crash

CHAPTER 7 REQUIREMENTS COMPILATION